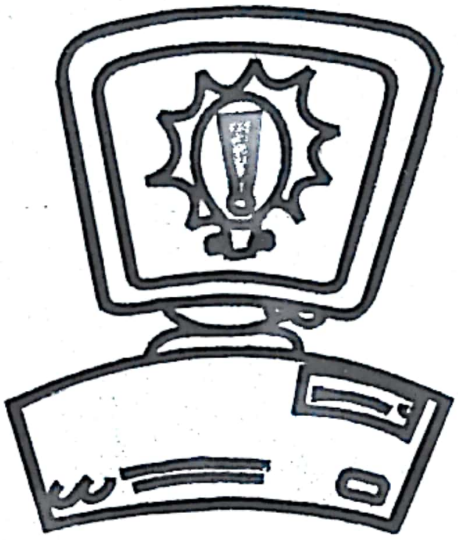
Claudia Botez Călin Neaga

Culegere de probleme de programare



Editura LORELEY

Scanned with OKEN Scanner

Universitatea Tehnică „Gh. Asachi" Iași
Facultatea de Automatică și Calculatoare

Părinților, cu toată dragostea - Călin

Claudia Botez

Călin Neaga

Culegere de probleme de programare (C, dBase)

– Editura Loreley –

Scanned with OKEN Scanner

Claudia Botez a absolvit Facultatea de Electrotehnică din Iaşi în anul 1966 şi Facultatea de Matematică-Mecanică din Iaşi în anul 1974. În prezent este conferenţiar în cadrul catedrei de calculatoare a Facultăţii de Automatică şi Calculatoare din Universitatea Tehnică Iaşi, activînd la disciplinele programarea calculatoarelor, baze de date, structuri de date şi algoritmi. Este coautor la cinci cărţi de specialitate.

Călin Neaga a absolvit Facultatea de Electrotehnică din Iaşi în anul 1987. În prezent este şef de lucrări în cadrul catedrei de calculatoare a Facultăţii de Automatică şi Calculatoare din Universitatea Tehnică Iaşi, activînd la disciplinele baze de date, ingineria programării, proiectarea bazelor de date. Este coautor la două cărţi de specialitate.

Scanned with OKEN Scanner

Introducere

...when it comes to creating messes,
no one beats a programmer.
Bertrand Meyer

Acest volum îşi propune să ofere o serie de soluţii pentru principalele probleme şi exerciţii propuse în două îndrumare de laborator apărute sub egida Facultăţii de Automatică şi Calculatoare din Universitatea Tehnică Iaşi (*Programarea calculatoarelor.Limbajul C-Implementarea TurboC++*, de Claudia Botez şi Călin Neaga, 1995, şi respectiv *Baze de date.dBASE IV-Îndrumar de laborator*, de Călin Neaga şi Petru Cașcaval, 1996).

Se adaugă în plus un capitol cu probleme suplimentare de programare în limbajul C.

Turbo C++ este o marcă înregistrată a firmei Borland International, iar dBase IV a fost creat de către firma Ashton-Tate, preluată ulterior de către Borland.

Problemele propuse se adresează în primul rînd studenţilor Facultăţii de Automatică şi Calculatoare şi ai Colegiului de Tehnică de Calcul din Iaşi, dar pot interesa şi alte categorii de potenţiali programatori.

Evident, soluţiile propuse nu sînt singurele posibile deci cu atît mai puţin cele mai bune. Am încercat să oferim cîte o soluţie pentru cele mai semnificative probleme din volumele menţionate, dar sîntem pe deplin convinşi că întotdeauna este loc de mai bine (neexcluzînd nici posibilitatea unor erori proprii de concepere şi/sau de rezolvare a problemelor).

La o bună parte din probleme, finisările (sau chiar variantele de rezolvare în sine) au rezultat din activitatea practică la laboratoare, aportul studenţilor fiind esenţial atît în clarificarea enunţurilor cît şi în propunerea unor soluţii mai inteligente decît cele pe care le-am oferit iniţial.

Orice observaţii sînt binevenite, de preferinţă prin e-mail:

- Claudia Botez la botez@cs.tuiasi.ro
- Călin Neaga la cneaga@cs.tuiasi.ro

Autorii

Scanned with OKEN Scanner

Capitolul I : Programare în C

A. Probleme din îndrumarul de laborator

Problema 3.3

Scrieţi un program care citeşte o secvenţă de întregi din intrare (folosiţi redirectarea *stdin*) şi determină suma elementelor având numerele de ordine 1, 3, 6, 10, 15, 21... .

Se observă (?!) că trebuie adunate elementele având numerele de ordine:

1
1 + 2
1 + 2 + 3
1 + 2 + 3 + 4
1 + 2 + 3 + 4 + 5
...

```
#include <stdio.h>
#include <conio.h>
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia inSerie returnează 1 dacă numărul transmis ca parametru face
parte din seria menţionată.
        Parametri:
                - x: numărul ce trebuie comparat cu 'pragurile' seriei;
        Returnează:
                - int: 1 dacă numărul transmis ca parametru face parte din serie.
                       altfel 0;
*/
int inSerie ( int x )
{
        int i = 1 ;
        int sumaInt = 0 ;
        while ( sumaInt < x )     {
                sumaInt += i++ ;
                if ( sumaInt == x )                    return 1 ;
                }
        return 0 ;
```


Scanned with OKEN Scanner

```
}
/*-------------------------------------------------------------------------------*/
int main ( void )
{
        int elem , nrOrdine=1 ;
        long suma = 0 ;
        clrscr() ;
        printf("\nIntroduceti numerele intregi ( se incheie cu Ctrl-Z ) :\n") ;
        while ( scanf("%d",&elem) != EOF )
                if ( inSerie ( nrOrdine++ ) )
                        suma += elem ;
        printf("\nRezultat final : %ld .",suma) ;
/*
        Un eventual getch() ar fi (aici) total neinspirat deoarece nu ar putea fi "dat"
atunci cînd s-a redirectat intrarea!
*/
        return 0 ;
}
```

Problema 3.4

Să se scrie un program care determină cel mai mare divizor comun al două numere întregi strict pozitive după algoritmul lui Euclid:

- se împarte numărul mai mare la cel mai mic atît timp cît împărțitorul este nenul;
- la fiecare pas, pînă cînd restul (se folosește operatorul modulo %) este zero, se înlocuiește deîmpărțitul cu împărțitorul și împărțitorul cu restul;
- ultimul împărțitor este c.m.m.d.c.;

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
/*-------------------------------------------------------------------------------*/
int main ( void )
{
        int nr1 , nr2 , rest , temp , cnt , nr1Mem , nr2Mem ;
```


Scanned with OKEN Scanner

```
        clrscr() ;
        printf("\nIntroduceti cele doua numere intregi : ") ;
        if ( scanf("%d %d", &nr1, &nr2) != 2 )  {
                printf("\n\aEroare introducere date !") ;
                exit(1) ;
                }
        if ( nr1 <= 0 || nr2 <= 0 )          {
                printf("\n\aCele două numere trebuie să fie strict pozitive !") ;
                exit(1) ;
                }
/*
        Se memorează cele doua numere în ordinea inițială (preventiv pentru
varianta interschimbării valorilor) - vor fi folosite la afișarea rezultatului final.
*/
        nr1Mem = nr1 ;
        nr2Mem = nr2 ;
// Se face interschimbarea valorilor dacă al doilea număr este mai mare decît primul.
        if ( nr2 > nr1 )  {
                temp = nr2 ;
                nr2  = nr1 ;
                nr1  = temp ;
                }
        while ( nr2 != 0 )          {
                if ( nr1 % nr2 == 0 )      break ;
                rest = nr1 % nr2 ;
                nr1 = nr2 ;
                nr2 = rest ;
                }
        cmmdc = nr2 ;
        if ( cmmdc == 1 )
                printf("\nNumerele %d si %d sînt prime .", nr1Mem, nr2Mem) ;
        else
                printf("\nC.M.M.D.C. al numerelor %d şi %d este  : %d .",
                                        nr1Mem, nr2Mem, cmmdc) ;
        printf("\n\n\t\tApăsaţi orice tastă pentru terminare ..") ;
        getch() ;
        return 0 ;
}
```

Problema 3.5

Scanned with OKEN Scanner

Refaceți programul precedent folosind funcția recursivă:

$$cmmdc(x, y) = \begin{cases} y & \text{dacă } x \% y = 0 \\ cmmdc(y, x\%y) & \text{altfel} \end{cases}$$

```
#include <stdio.h>
#include <conio.h>
#include <process.h>
/*-------------------------------------------------------------------------------*/
/*
        Funcția cmmdc determină recursiv cel mai mare divizor comun al celor
două numere întregi transmise ca parametri.
        Parametri :
                - nr1: numerele al căror CMMDC se determină:
                - nr2
        Returnează:
                - int: valoarea CMMDC:
*/
int cmmdc ( int nr1 , int nr2 )
{
        if ( nr2 == 0 )
                return ( nr1 ) ;
        else
                cmmdc ( nr2 , nr1 % nr2 ) ;
}
/*
        Compilatorul nu va sesiza că funcția returnează (în final) ceva și se va emite
avertismentul:
                Function should return a value in function cmmdc
*/
/*-------------------------------------------------------------------------------*/
int main ( void )
{
        int a , b ;
        clrscr() ;
        printf("Introduceti cele doua numere intregi : ") ;
        if ( scanf ( "%d %d", &a, &b ) = 2 )    {
```


Scanned with OKEN Scanner

```
            printf("\n\aEroare introducere numere !") ;
            exit(1) ;
            }
    if ( a <= 0 || b <= 0)      {
            printf("\n\aNumerele trebuie sa fie strict pozitive !") ;
            exit(1) ;
            }
    printf("\nC.m.m.d.c. al numerelor %d si %d este : %d.",
                    a, b, cmmdc ( a, b ) ) ;
    printf("\n\n\t\tApăsaţi orice tastă pentru terminare...") ;
    getch() ;
    return 0 ;
}
```

Problema 4.1

Să se scrie un program care crează un pătrat magic de dimensiune prestabilită.

Un pătrat magic (de latură n) are proprietatea că include în locaţiile sale toate numerele întregi din intervalul $1 \div n^2$ şi sumele numerelor de pe fiecare linie, fiecare coloană sau fiecare diagonală sînt egale.

De exemplu:

6	**1**	**8**
7	**5**	**3**
2	**9**	**4**

Pentru crearea unui pătrat magic (de dimensiune impară!) se va folosi regula lui *H.Coxeter* ([3], pag. 23): se porneşte cu numărul 1 în mijlocul primei linii, apoi se merge la fiecare pas cu o linie în sus şi cu o coloană la stînga, scriind numerele în ordine crescătoare în locaţiile libere.

De pe prima linie se va trece pe ultima, iar din prima coloană în ultima.

Dacă o locaţie ce ar urma în acest mod este deja ocupată, se trece la cea situată cu o linie în jos (pe aceeaşi coloană).

Se vor afişa atît pătratul rezultat cît şi toate sumele ce ar trebui să coincidă.

(a) varianta cu tablouri

```
#include <stdio.h>
```


Scanned with OKEN Scanner

```
#include <conio.h>
#include <dos.h>
#include <process.h>
#include <alloc.h>

#define DIM 15
int  patrat[DIM][DIM] ;
int  dimensPatrat;
/*-------------------------------------------------------------------------------------------*/
/*
```

Funcţia **sumaLinie** calculează şi returnează suma elementelor de pe linia pătratului avînd ca indice valoarea transmisă ca parametru.

Parametri:

- *nrLinie*: indicele liniei dorite;

Returnează:

- *long*: suma elementelor de pe linia respectivă;

```
*/
long sumaLinie ( int nrLinie )
{
 int nrColoana ;
 long sum = 0 ;
 for ( nrColoana=0; nrColoana<dimensPatrat; nrColoana++)
       sum += patrat[nrLinie][nrColoana] ;
 return sum ;
}
/*-------------------------------------------------------------------------------------------*/
/*
```

Funcţia **sumaColoana** calculează şi returnează suma elementelor de pe coloana pătratului avînd ca indice valoarea transmisă ca parametru.

Parametri:

- *nrColoana*: indicele coloanei dorite;

Returnează :

- *long*: suma elementelor de pe coloana respectivă;

```
*/
long sumaColoana ( int nrColoana )
{
 int nrLinie ;
 long sum = 0 ;
 for ( nrLinie=0; nrLinie<dimensPatrat; nrLinie++)
       sum += patrat[nrLinie][nrColoana] ;
 return sum ;
```


Scanned with OKEN Scanner

```
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia sumaDiagonala1 calculează şi returnează suma elementelor de pe
diagonala principală a pătratului.
        Parametri:
                - void;
        Returnează:
                - long: suma elementelor de pe diagonala respectivă;
*/
long sumaDiagonala1 ( void )
{
 int nrLinCol ;
 long sum = 0 ;
 for ( nrLinCol=0; nrLinCol<dimensPatrat; nrLinCol++)
        sum += patrat[nrLinCol][nrLinCol] ;
 return sum ;
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia sumaDiagonala2 calculează şi returnează sumă elementelor de pe
diagonala secundară a pătratului.
        Parametri:
                - void;
        Returnează:
                - long: suma elementelor de pe diagonala respectivă;
*/
long sumaDiagonala2 ( void )
{
 int nrColLin ;
 long sum = 0 ;
 for ( nrColLin =dimensPatrat-1; nrColLin>=0; nrColLin--)
        sum += patrat[nrColLin][nrColLin] ;
 return sum ;
}
/*-------------------------------------------------------------------------------------*/
int main ( void )
{
 int linia , coloana , nouaLinie , nouaColoana , valoareCurenta ;
 clrscr() ;
 printf ( "Dimensiunea patratului ( numar impar <= 17 ) : " ) ;
```


Scanned with OKEN Scanner

```
        sum += *(patrat + nrLinie * dimensPatrat + nrColoana) ;
  return sum ;
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia sumaColoana calculează şi returnează suma elementelor de pe
coloana pătratului avînd ca indice valoarea transmisă ca parametru.
        Parametri:
                - nrColoana: indicele coloanei dorite;
        Returnează:
                - long: suma elementelor de pe coloana respectivă;
*/
long sumaColoana ( int nrColoana )
{
  int nrLinie ;
  long sum = 0 ;
  for ( nrLinie=0; nrLinie<dimensPatrat; nrLinie++)
        sum += *(patrat + nrLinie * dimensPatrat + nrColoana)  ;
  return sum ;
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia sumaDiagonala1 calculează şi returnează suma elementelor de pe
diagonala principală a pătratului.
        Parametri:
                - void;
        Returnează:
                - long: suma elementelor de pe diagonala respectivă;
*/
long sumaDiagonala1 ( void )
{
  int nrLinCol ;
  long sum = 0 ;
  for ( nrLinCol=0; nrLinCol<dimensPatrat; nrLinCol++)
        sum += *(patrat + nrLinCol * dimensPatrat + nrLinCol) ;
  return sum ;
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia sumaDiagonala2 calculează şi returnează suma elementelor de pe
diagonala secundară a pătratului.
```


Scanned with OKEN Scanner

```
        Parametri:
                - void;
        Returnează:
                - long: suma elementelor de pe diagonala respectivă;
*/
long sumaDiagonala2 ( void )
{
 int nrColLin ;
 long sum = 0 ;
 for ( nrColLin=dimensPatrat-1; nrColLin>=0; nrColLin--)
        sum += *(patrat + nrColLin * dimensPatrat + nrColLin) ;
 return sum ;
}
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 int linia , coloana , nouaLinie , nouaColoana , valCurenta ;
 clrscr() ;
 printf("Dimensiunea patratului : ") ;
 scanf ( "%d", &dimensPatrat ) ;
 if ( dimensPatrat <= 1 )          {
        printf("\aNu se accepta decit valori mai mari decit 1 !") ;
        exit(1) ;
        }
 if ( dimensPatrat % 2 ==0 )      {
        printf("\aTrebuie o dimensiune impara !") ;
        exit(1) ;
        }
if ( (patrat = (int *) malloc ( dimensPatrat * dimensPatrat * sizeof(int) ) ) == NULL)
        {
        printf("Eroare alocare memorie ...") ;
        exit(1) ;
        }
// Se inițializează toate celulele cu valoarea 0.
 for (linia=0; linia<dimensPatrat; linia++)
        for (coloana=0; coloana<dimensPatrat; coloana++)
                *(patrat + linia*dimensPatrat + coloana) = 0 ;
 linia = 0 ;
 coloana = ( dimensPatrat - 1 ) / 2 ;      // se fixează poziția de pornire
// linia și coloana reprezintă linia și respectiv coloana poziției curente.
 *(patrat + linia*dimensPatrat + coloana) = 1 ;
```


Scanned with OKEN Scanner

```
valCurenta = 2 ;          // valoarea ce se scrie la următorul pas
while ( valCurenta <= dimensPatrat * dimensPatrat )     {
  if ( linia-1 < 0)
        nouaLinie = dimensPatrat - 1 ;
  else
        nouaLinie = linia - 1 ;
  if ( coloana-1 < 0)
        nouaColoana = dimensPatrat - 1 ;
  else
        nouaColoana = coloana - 1 ;
// nouaLinie şi nouaColoana reprezintă următoarea poziţie în varianta implicită.
    if ( *(patrat + nouaLinie*dimensPatrat + nouaColoana) != 0)
                        // celula era ocupată
        linia = (linia+1) % dimensPatrat ;
    else                {
        linia = nouaLinie ;
        if ( coloana-1 < 0)             // se trece de pe prima coloană pe ultima
                coloana = dimensPatrat - 1 ;
        else
                coloana-- ;
        }
// S-au actualizat linia şi coloana în vederea scrierii următorului număr.
    *(patrat + linia*dimensPatrat + coloana) = valCurenta++ ;
    }
 for (linia=0; linia<dimensPatrat; linia++)                {
        for (coloana=0; coloana<dimensPatrat; coloana++)
                printf("%3d   ",*(patrat + linia*dimensPatrat + coloana)) ;
        printf("\n") ;
        }
printf("\nApasati orice tasta pentru a trece la afisarea sumelor...\n") ;
getch() ;
for (linia=0; linia<dimensPatrat; linia++)
        printf ( "Linia   %d  : %ld\n", linia, sumaLinie(linia) ) ;
for (coloana=0; coloana<dimensPatrat; coloana++)
        printf ( "Coloana %d  : %ld\n", coloana, sumaColoana(coloana)) ;
printf ( "Diagonala 1 : %ld\n", sumaDiagonala1() ) ;
printf ( "Diagonala 2 : %ld\n", sumaDiagonala2() ) ;
free(patrat) ;
printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
getch() ;
return 0 ;
```


Scanned with OKEN Scanner

}

Problema 4.2

Să se scrie un program care citeşte un tablou de maximum 30 de numere întregi de la tastatură, le sortează (ordonare crescătoare) prin metoda bulelor şi afişează rezultatul.

Metoda constă în compararea tuturor perechilor a_i-a_{i+1}, $0 \le i < N-1$ (*N*:numărul elementelor), şi inversarea lor dacă este cazul. Operaţia se reia atît timp cît s-a făcut vreo inversiune.

Să se facă o variantă a programului la care se va indica prin dialog dacă ordonarea să se facă crescător sau descrescător.

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia sortareBule sortează prin metoda bulelor elementele unui tablou de
întregi.
        Parametri :
          - x: numele tabloului;
          - nrElemente: numărul de elemente ale tabloului;
          - tipSortare: varianta de sortare (1 - crescător, respectiv 0 - descrescător);
        Returnează :
          - void;
*/
void sortareBule ( int x[] , int nrElemente , int tipSortare )
{
 int i , temporar , inversiuni ;
 do     {
   inversiuni = 0 ;
   for ( i=0; i < nrElemente - 1 ; i++)
        if ( tipSortare ? ( x[i] > x[i+1] ) : ( x[i] < x[i+1] ) )        {
                temporar = x[i] ;
                x[i] = x[i+1] ;
                x[i+1] = temporar ;
                inversiuni = 1 ;
                }
        } while ( inversiuni != 0 ) ;   // sau while ( inversiuni )
```


Scanned with OKEN Scanner

```
}
/*-------------------------------------------------------------------------------*/
int main ( void )
{ .
 int i , a[30] , x , dimTablou , varSortare ;
 clrscr() ;
 printf("\n Introduceti numerele ( maxim 30 , încheiere cu Ctrl-Z\nsau orice alt"
        " caracter decît cifrele zecimale ) :\n\n") ;
 for ( i=0; i<30; ) {
   if ( scanf("%d",&x) != 1 )              break ;
   a[i++] = x ;
   }
 dimTablou = i ;          // numărul de elemente efectiv introduse
 printf("\n Crescător / descrescător ( C/D ) ? ") :
                  // Se aşteaptă pînă se indică explicit 'C' sau 'D'.
 do     {
   varSortare = getch() ;
   varSortare = tolower ( varSortare ) ;
   }     while ( varSortare != 'c' && varSortare != 'd' )    ;
 putch ( toupper ( varSortare ) ) ;
                  // getch() NU face şi afişarea în ecou a caracterului tastat !
 sortareBule ( a , dimTablou , varSortare == 'c' ? 1 : 0 ) ;
 printf("\n\n\nRezultatul sortării %s :\n",
                varSortare == 'c' ? "crescătoare" : "descrescătoare" ) ;
 for (i=0; i<dimTablou; i++)
        printf("%6d%c", a[i], ( (i+1) % 6 ==0 ) ? '\n' : ' ') ;
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 4.3

Să se scrie un program care determină şi afişează toate numerele prime inferioare unui număr dat (prestabilit - *n*), folosind ciurul lui Eratostene.

Metoda constă din eliminarea succesivă a numerelor neprime:

(a) Se elimină numărul 1 (prin definiţie nu este prim).

Scanned with OKEN Scanner

(b) Apoi se caută (pornind de ultimul număr prim considerat - inițial 1) primul număr neeliminat (acesta este prim) și se elimină toți multiplii săi (din intervalul 1÷n).

(c) Se repetă pasul (b) pînă cînd numărul prim considerat este mai mare decît $n^{1/2}$ (radical din *n*).

Pentru radical se va folosi funcția (cu prototipul în MATH.H):

double **sqrt** (double x) ;

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define DIM 1000
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 int i , j , k ,l ;
 int num[DIM+1] ; // se va ignora elementul num[0]
 for (i=1; i<=DIM; i++)
   num[i]=1 ;   // elementele num[] vor deveni 0 pentru numerele neprime
 num[1] = 0 ;   // pasul (a)
 j = 2 ;
 do     {
   for ( k=j; k <= DIM; k++)
        if ( num[k] == 1)
              break ;  // s-a ajuns la următorul numar prim
        for ( l=2; k*l <= DIM; l++)
              num[k*l] = 0 ;
             // se elimină ( numere neprime ) toți multiplii săi mai mici decît DIM
        j++ ;
        } while ( k < sqrt(DIM) ) ;
 clrscr() ;
 printf("\nNumerele prime intre 1 si %d sint :\n",DIM) ;
 l = 0 ;
 for ( i=1; i<=DIM; i++)        {
        if ( num[i] )     {
              printf("%4d   ",i) ;
              l++ ;
              if (l==8)        {
                    printf("\n") ;
```


Scanned with OKEN Scanner

```
                    l = 0 ;
                    }
                }
        }
printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
getch() ;
return 0 ;
}
```

Problema 6.3

Rezolvați problema lui *Josephus*: sînt N oameni așezați în cerc (avînd numerele de ordine 1÷N) și la fiecare pas se elimină al K-lea, pînă cînd rămîne unul singur.

La primul tur se începe numărătoarea de la primul om, apoi de la cel (neeliminat) imediat următor celui eliminat la pasul curent.

Se cere numărul de ordine pentru "supraviețuitor".

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#include <stdlib.h>
/*-------------------------------------------------------------------------------*/
int main ( void )
{
 int nrPersoane, kEliminare, *persoane, i, j, nrRamase, nrCurent;
 clrscr() ;
 printf("\nIntroduceti valorile N si K :") ;
 scanf ( "%d%d", &nrPersoane, &kEliminare) ;
 if ( nrPersoane<2 || kEliminare<2 )      {
        printf("\n\aTrebuie ca ambele valori sa fie de minimum 2 ...") ;
        exit(1) ;
        }
 if ( ( persoane = (int *) calloc ( nrPersoane, sizeof(int) ) ) == NULL)      {
        printf("\n\aEroare alocare memorie ...") ;
        exit(1) ;
        }
                // calloc() face și inițializarea cu 0.
 nrRamase = nrPersoane ; // numărul de persoane rămase ( neeliminate ).
 nrCurent = 0 ;
```


Scanned with OKEN Scanner

```
while ( nrRamase > 1 ){
        for ( i=nrCurent , j=0; ; i = (i+1) % nrPersoane )
                        // (i+1)%... asigură trecerea de la ultimul la primul
                if ( *(persoane+i) == 0 ) {
                        j++ ;
                        if ( j == kEliminare )              break ;
                        }
                         // se "numără" K persoane neeliminate
        *(persoane+i) = nrPersoane - nrRamase + 1 ;
           // Se face eliminarea prin completarea cu numărul de ordine al eliminării
           // curente ( pentru testare ).
        nrRamase-- ;
        for ( j = (i+1) % nrPersoane;*(persoane+j) != 0; j = (j+1) % nrPersoane )
                ;
         // Se trece la prima persoană neeliminată - se actualizează variabila pentru
         // numărul curent.
        nrCurent = j ;
        }
// Se determină indicele corespunzător persoanei neeliminate:
 for (i=0; i<nrPersoane; i++)
        if ( *(persoane+i) == 0 )          break ;
 printf("\nRamine : %d",i+1) ;
/*
  Pentru testare se poate încerca mai întîi:
        for (i=0; i<nrPersoane; i++)
                if ( *(persoane+i) == 0 )
                        printf("\nRamine : %d",i+1) ;
        şi ar trebui să rămînă (i.e.: să aibă valoarea 0) una singură!
*/
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0;
}
```

Problema 7.1

Să se scrie un program care citeşte de la tastatură două tablouri de numere întregi, le sortează pe fiecare, după care le interclasează şi afişează rezultatul.

Programul principal, funcţia de sortare şi funcţia de interclasare vor fi plasate în fişiere compilate separat.

Scanned with OKEN Scanner

Pentru interclasare se va folosi algoritmul (pseudocod: din [5], pag. 266):

```
procedure INTER ( A, m, B, n, C )
array A(m), B(n), C(m+n)
  i ← 1 ; j ← 1 ; k ← 0 ;
   while i ≤ m  ∧  j ≤ n
        k ← k + 1
        if a_i < b_j  then  c_k ← a_i ;  i ← i + 1
                    else   c_k ← b_j ;  j ← j + 1
        endif
   repeat
   if  i > m   then       for  l = j , n :
                                  k ← k + 1 ;  c_k ← b_l
                          repeat
                 else     for  l = i , m :
                                  k ← k + 1 ;  c_k ← a_l
                          repeat
   endif
return
end
( ← : asignare ;
  for l = i , m     ≡     for ( l=i;, l<=m; l++ )   ;
  nu uitați că indicele primului element al unui tablou în C este 0 ! )
```

Se prezintă soluția sub forma unui fișier unic - pentru a ne conforma enunțului, am crea trei fișiere separate, reunindu-le într-un fișier proiect.

```
#include <stdio.h>
#include <conio.h>
#define M 10
#define N 15
/*-------------------------------------------------------------------------------------------*/
/*
```

Funcția **sortareBuleCresc** este o variantă simplificată a celei prezentate la problema 6.3: se ordonează crescător (prin metoda bulelor) elementele unui tablou de întregi.

Parametri:

- *x*: numele tabloului ce se sortează;
- *dimTablou*: numărul de elemente ale tabloului;

Returnează:

- *void*;

Scanned with OKEN Scanner

```
*/
void sortareBuleCresc ( int x[] , int dimTablou )
{
 int i , temp , inv ;
 do      {
         inv = 0 ;
         for ( i=0; i<dimTablou-1; i++)
                 if ( x[i] > x[i+1] )          {
                         temp = x[i] ;
                         x[i] = x[i+1] ;
                         x[i+1] = temp ;
                         inv = 1 ;
                         }
                 } while ( inv!=0 ) ;
}
/*-----------------------------------------------------------------------------------------*/
/*
```

Funcția **interclasareCresc** realizează interclasarea a două tabele de întregi (este obligatoriu ca acestea să fie deja sortate!), depunînd rezultatul într-un al treilea (s-a alocat spațiul necesar pentru acesta în funcția *main*).

Parametri :

- *a*: numele primului tablou;
- *dimA*: numărul de elemente ale primului tablou;
- *b*: numele celui de-al doilea tablou;
- *dimB*: numărul de elemente ale celui de-al doilea tablou;
- *c*: numele tabloului în care se depune rezultatul interclasării (va avea *dimA* + *dimB* elemente);

Returnează:

- *void*;

```
*/
void interclasareCresc ( int a[] , int dimA , int b[] , int dimB , int c[] )
{
/*
```

Prin convenție, dacă aceeași valoare apare în ambele tablouri, va fi scrisă de mai multe ori în tabloul final .

```
*/
 int i , j , k ;
 i = j = k = 0 ;
 while ( i < dimA && j < dimB )          {
         if ( a[i] < b[j] )
                 c[k++] = a[i++] ;
```


Scanned with OKEN Scanner

```
        else if ( a[i] > b[j] )
                c[k++] = b[j++] ;
        else
                c[k++] = a[i++] ;
        }
 if ( i == dimA )
        while ( j < dimB )
                c[k++] = b[j++] ;
 else
        while ( i < dimA )
                c[k++] = a[i++] ;
}
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 int a[M] , b[N] , c[M+N] ;
 int i , nrElemA , nrElemB , x ;
 clrscr() ;
/*
        Se încheie citirea pentru mai puțin de 10 elemente prin orice caracter diferit
de cifrele zecimale (și eventualul semn).
*/
 printf("\nIntroduceti elementele tabloului A :\n") ;
 for (i=0; i<M; i++)     {
        if ( scanf("%d",&x)!=1 )                    break ;
        a[i] = x ;
        }
 nrElemA = i ; // numărul de elemente ale tabelului A (posibil mai mic decît M)
 fflush(stdin) ; // Se elimină caracterele necitite din intrare (Ctrl-Z sau inacceptabile
                 // la citirea numerelor).
 printf("\nIntroduceti elementele tabloului B :\n") ;
 for (i=0; i<N; i++)     {
        if ( scanf("%d",&x)!=1 )                    break ;
        b[i] = x ;
        }
 nrElemB = i ;  // numărul de elemente ale tabelului B (posibil mai mic decît N)
 fflush(stdin) ;
 sortareBuleCresc ( a , nrElemA ) ;
 sortareBuleCresc ( b , nrElemB ) ;
 interclasareCresc ( a , nrElemA , b , nrElemB , c ) ;
 printf("\n\nRezultatele interclasării :\n"
```


Scanned with OKEN Scanner

```
 for ( i=0; i < nrElemA + nrElemB ; i++)
        printf("%6d%c", c[i], ((i+1)%7==0) ? '\n' : ' ' ) ;
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 7.2

Să se scrie un program care plasează N dame (în sensul de regine) pe o tablă de şah N*N astfel încît să nu existe două dame pe aceeaşi linie, coloană sau diagonală.

Se va adapta şi codifica procedura ([5], pag. 105):

```
procedure DAME ( n )
integer x ( n )
x1 ← 0 ; k ← 1
  while  k > 0
         v ← 0
            while  xk  ≤ n - 1
                xk ← xk + 1
                call POSD ( x, k, v )
                   if  v = 1    then exit
                   endif
             repeat
             if  v = 0  then  k ← k - 1
                      else        if  k = n  then write x
                                              else k ← k + 1 ; xk ← 0
                                  endif
             endif
    repeat
return
end

procedure POSD ( x, k, v )
integer x ( k )
v ← 0
   for i = 1 , k - 1
          if  xk = xi  ∨  k - i = | xk - xi |   then return
```


Scanned with OKEN Scanner

```
            endif
    repeat
v ← 1
return
end
```

Atenție: *exit* din pseudocod corespunde instrucțiunii C *break* !

```
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <ctype.h>
#define N 8
        // dimensiunea tablei de şah (se pot încerca şi alte valori)
int nrSolutii = 0 ; // se memorează cîte posibile soluții s-au găsit
/*-------------------------------------------------------------------------------------------*/
/*
Functia damaPosibila determină dacă poate fi amplasată o damă în poziția indicată.
        Parametri:
        - x: numele tabloului în care se memorează pozițiile (elementul x[i] are
             valoarea corespunzătoare coloanei pe care se pune dama din linia i);
        Returnează:
                - int: 1 dacă poziția este acceptabilă, altfel 0;
*/
int damaPosibila ( int x[] , int k )
{
 int i ;
 for ( i=0; i<k; i++)
        if ( x[k]==x[i] || k-i==abs(x[k]-x[i]) )          return 0 ;
 return 1 ;
}
/*-------------------------------------------------------------------------------------------*/
/*
        Funcția linie trasează o 'linie' pentru a 'desena' (sumar) tabla de şah.
        Parametri :
                - void;
        Returnează:
                - void;
*/
void linie ( void )
```


Scanned with OKEN Scanner

```
{
 int i ;
 for (i=0; i<N; i++)
        printf("+---") ;
 printf("+\n") ;
}
/*-------------------------------------------------------------------------------------------*/
/*
        Funcția afisareSolutie 'desenează' rezultatul problemei.
        Parametri:
                - x: tabloul cu valorile corespunzătoare (s-a detaliat la prima funcție);
        Returnează:
                - void;
        Modifică variabila globală nrSolutii.
*/
void afisareSolutie ( int x[] )
{
  int i , k ;
  nrSolutii++ ;
  for ( i=0; i<N; i++)     {
          linie() ;
          for (k=0; k<N; k++)
                  printf("| %c ", (x[k]==i) ? '*' : ' ' ) ;
          printf("|\n") ;
          }
   linie() ;
   printf("\n\nDoriți şi altă soluție ( D/N ) ? ") ;
   do     {
          k = getch() ;
          if ( k==0x00 )
                  getch() ;          // tastele speciale generează o pereche de coduri
          k = tolower(k) ;
          } while ( k!='d' && k!='n' ) ;
    printf("\n\n") ;
    if (k=='n')       exit(1) ;
    else              return ;
  }
  /*-------------------------------------------------------------------------------------------*/
  int main ( void )
  {
   int tabla[N] ;
```


Scanned with OKEN Scanner

```
/*
        Se alege un tablou unidimensional - cîte un element pentru fiecare linie (sau
coloană), valoarea elementului fiind coloana (respectiv linia) pe care s-a aşezat dama
corespunzătoare acelei linii (coloane).
*/
 int linCol ;
 int v ;
 clrscr() ;
 linCol = 0 ;
 tabla[0] = -1 ;
 while ( linCol >= 0 )    {
        v = 0 ;
        while ( tabla[linCol]<N )          {
                tabla[linCol]++ ;
                v = damaPosibila ( tabla , linCol ) ;
                if ( v==1 )                break ;
                }
        if ( v==0 || tabla[linCol]>=N )
                linCol-- ;
        else    {
                if ( linCol==N-1 )                        afisareSolutie(tabla) ;
                else                {
                        linCol++ ;
                        tabla[linCol] = -1 ;
                        }
                }
        }
 printf("\nS-au epuizat toate cele %d solutii !",nrSolutii) ;
                // unele soluţii sînt de fapt identice dar rotite cu 90°-180°-270° !
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 7.3

Să se încerce un program pentru parcurgerea unei table de şah prin toate pătratele, pornind dintr-un colţ şi mutînd un cal, fără a trece de mai multe ori prin aceeaşi poziţie.

Scanned with OKEN Scanner

De la dimensiunea tablei de la 8 în sus ar fi mai util să se aplice metoda *J.C.Warnsdorff* ([3], pag. 63): calul trebuie mutat spre unul din pătratele de la care sînt cele mai puține ieşiri către pătrate încă netraversate.

Atenție - soluția propusă "rateaza" solutia pentru tabla de 7x7 !

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#define N 20
int dimTabla ,          // dimensiunea tablei
  a[N][N] ,             //* marchează daca s-a trecut (1) sau nu (0) prin a[i][j]
  memMut[N][N] ;  // marchează la a cîta mutare s-a trecut prin pătratul respectiv
int contor ;                                  /* numărul de mutari efectuate */
int pasi[8][2] = { { -2, -1 } ,               /* cele opt mutări teoretic posibile */
                   { -1, -2 } ,               /* modificarea liniei / coloanei */
                   {  1, -2 } ,
                   {  2, -1 } ,
                   {  2,  1 } ,
                   {  1,  2 } ,
                   { -1,  2 } ,
                   { -2,  1 } } ;
/*-------------------------------------------------------------------------------------*/
/*
        Funcția linie trasează o 'linie' pentru a 'desena' (sumar) tabla de şah.
        Parametri:
                - void;
        Returnează:
                - void;
*/
void linie ( void )
{
  int x ;
  for (x=0; x<dimTabla; x++)
        printf("+----") ;
  printf("+\n") ;
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcția afisareSolutie 'desenează' rezultatul problemei.
        Parametri:
```


Scanned with OKEN Scanner

```
                - x: tabloul cu valorile corespunzătoare;
        Returnează:
                - void;
*/
void afisareSolutie ( void )
{
 int i , k ;
 for (i=0; i<dimTabla; i++)  {
        linie() ;
        for (k=0; k<dimTabla; k++)
                printf("|%3d ",memMut[i][k]) ;
        printf("|\n") ;
        }
 linie() ;
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcția numarPasi determină cîte mutări sînt posibile din pătratul ale cărui
coordonate se transmit ca parametri.
        Parametri:
                - linia: coordonatele pătratului de pe tablă;
                - coloana
        Returnează:
                - int: numărul de mutări posibile (0 dacă pătratul este deja ocupat);
*/
int numarPasi ( int linia , int coloana )
{
 int posib ,                    /* numărul de posibilități */
    i ;
 i = posib = 0 ;
 if ( 0 <= linia && linia < dimTabla && 0 <= coloana &&
    coloana < dimTabla && !a[linia][coloana] ) {
        if ( contor == dimTabla * dimTabla - 1 )
                posib = 1 ;
        else
          do    {       // se testează fiecare variantă dacă este posibilă
            if ( (0 <= linia + pasi[i][0]) && (linia + pasi[i][0] < dimTabla) &&
                (0 <= coloana + pasi[i][1]) && (coloana + pasi[i][1] < dimTabla)
                 && !a[linia+pasi[i][0]][coloana+pasi[i][1]] )
                        posib++ ;
            i++ ;
```


Scanned with OKEN Scanner

```
            }  while ( i < 8 ) ;
        }
 return posib ;
}
/*-----------------------------------------------------------------------------------------------*/
/*
        Funcția mutareCal efectuează urmatoarea mutare.
        Parametri:
                - linia: linia corespunzătoare poziției curente;
                - coloana: coloana corespunzătoare poziției curente;
        Returnează:
                - void;
        Modifică variabila globală contor.
*/
void mutareCal ( int linia , int coloana )
{
 int nrPasi ,            /* numărul de paşi încercați */
    nouaLinie ,          /* linia pătratului în care se face următoarea mutare */
    nouaColoana ,        /* coloana   "    "   "   "   "        "        "   */
    mutariPosibile ,     /* cîte mutări vor fi posibile din nouaLinie][nouaColoana] */
    variantePosibile ;   /*   "    "    "   "    "     după pasul curent */
 contor++ ;
 a[linia][coloana] = 1 ;
 memMut[linia][coloana] = contor ;
 if ( contor < dimTabla * dimTabla )     {
    nrPasi = 0 ;
    mutariPosibile = 8 ;
    do    {
      variantePosibile = numarPasi ( linia + pasi[nrPasi][0] ,
                                     coloana + pasi[nrPasi][1] );
      if ( variantePosibile < mutariPosibile && variantePosibile > 0 )      {
   /* nouaLinie , nouaColoana vor fi coordonatele pentru următoarea mutare */
        mutariPosibile = variantePosibile ;
        nouaLinie = linia + pasi[nrPasi][0] ;
        nouaColoana = coloana + pasi[nrPasi][1] ;
        }
      nrPasi++ ;
      }  while ( nrPasi<8 && mutariPosibile !=1 ) ;
    if ( nrPasi == 8 && mutariPosibile == 8 )
                        /* nu mai există nici o mutare posibilă */
        return ;
```


Scanned with OKEN Scanner

```
    mutareCal ( nouaLinie , nouaColoana ) ;
    }
 else    {                  /* turul s-a încheiat cu succes */
         afisareSolutie() ;
         printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
         getch() ;
         exit(0) ;
         }
}
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 clrscr() ;
 printf("Dimensiunea tablei : ") ;
 scanf("%d",&dimTabla) ;
 if (dimTabla<3 || dimTabla>19)          {
         printf("\n\aDimensiunea trebuie să fie între 3 şi 19 ...") ;
         getch() ;
         exit(0) ;
         }
/*
 if (dimTabla==7)        {
         printf("\n\aAceastă variantă ratează soluţia pentru 7x7 ...") ;
         getch() ;
         exit(1) ;
         }
*/
 contor = 0 ;
 mutareCal ( 0 , 0 ) ;     // se porneşte din colţul din stînga-sus
 printf("\n\aNu s-a găsit vreo soluţie - apăsaţi orice tastă ...") ;
 getch() ;
 return 0 ;
}
```

Problema 8.1

Să se scrie un program care afişează la fiecare apăsare a unei taste două numere corespunzînd aruncării unei perechi de zaruri.

```
#include <stdio.h>
```


Scanned with OKEN Scanner

```
#include <stdlib.h>
#include <conio.h>
#include <time.h>
#define ESC 0x1b
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 int c ;
 clrscr() ;
 randomize() ;
 printf("\t  La apasarea fiecarei taste se va afisa o pereche de\n"
        "\tnumere corespunzind aruncarii unei perechi de zaruri.\n"
        "\t        ( se intrerupe prin tasta ESC )\n\n" ) ;
 while ( (c=getch()) != ESC )     {
        if (c==0x00)          // tastele speciale generează o pereche de coduri !
                getch() ;   // se elimină din intrare al doilea octet
        printf ( "%d - %d\n", 1 + random(6), 1 + random(6) ) ;
        }
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 8.2

Să se scrie un program care simulează o tragere excepțională LOTO. Sînt două faze: în prima se efectuează patru extrageri de cîte 12 numere din 90, iar în a doua fază (după completarea urnei cu toate cele 90 de numere) se fac 3 extrageri de cîte 8 numere, fără să se mai reintroducă bilele extrase înapoi în urnă după fiecare extragere.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 int extragerea1[12] ,
     extragerea2[24] ,  /* se foloseste un tablou unidimensional pentru a simula un
```


Scanned with OKEN Scanner

```
                                tablou bidimensional de 3 linii * 8 coloane */
    i , k , nrExtragere  ;
clrscr() ;
randomize() ;
printf("\n\n\nFAZA I   :\n\n") ;
for ( nrExtragere=1; nrExtragere<4; nrExtragere++)      {
  for ( i=0; i<12; i++)  {
     extragerea1[i] = 1 + random(90) ;
     for ( k=0; k<i; k++)
        while (extragerea1[k] == extragerea1[i])
           extragerea1[i] = 1 + random(90) ;
        // Se repetă extragerea unui nou număr atît timp cît coincide cu numărul
        // curent dintre cele deja extrase.
   }
 printf("\nExtragerea %d   : ",nrExtragere ) ;
 for (k=0; k<12; k++)
        printf("%4d",extragerea1[k]) ;
 }
 printf("\n\n\n\n\nFAZA II   :\n\n") ;
 for ( nrExtragere=0; nrExtragere<3; nrExtragere++)      {
    for ( i=0; i<8; i++)   {
        extragerea2[nrExtragere*8+i] = 1 + random(90) ;
        for ( k=0; k<nrExtragere*8+i; k++)
          while ( extragerea2[k] == extragerea2[nrExtragere*8+i] )
                extragerea2[nrExtragere*8+i] = 1 + random(90) ;
         }
    printf("\nExtragerea %d   : ",nrExtragere+1) ;
    for (k=0; k<8; k++)
        printf("%4d",extragerea2[nrExtragere*8+k]) ;
    }
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 8.3

Să se scrie un program care cere stabilirea unei parole de la tastatură şi apoi introducerea unei parole, face compararea şi afişează rezultatul.

Scanned with OKEN Scanner

La stabilirea parolei (maximum 8 caractere) se vor accepta numai litere, cifre şi caracterul '_' (*underscore*), putîndu-se folosi pentru editare tasta *Backspace*.

Nu se vor afişa în ecou caracterele introduse pentru parolă, ci un caracter grafic ASCII (de exemplu, *0xFE*), pentru a marca numărul de caractere deja introduse.

Se va cere (eventual) confirmarea parolei stabilite, cu un mesaj de eroare pentru existenţa vreunei neconcordanţe.

La ştergerea (din programul propriu) a caracterului aflat pe ecran în stînga cursorului se va folosi secvenţa :

<backspace><space><backspace> .

Pentru compararea a două şiruri de caractere se poate folosi funcţia:

int **strcmp** (const char *s1, const char *s2) ;

cu prototipul în **STRING.H**, care va returna 0 în caz de coincidenţă.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
void citesteParola ( char [] , int ) ;
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 char pass[9] ;
 char *test ;
 clrscr() ;
 printf("\nIntroduceti parola ( maximum 8 caractere inainte de <Enter > ) :\n") ;
 citesteParola ( pass , 8 ) ;
 printf("\n\nAti introdus parola : %s \n",pass) ;
 test = getpass ( "\nIntroduceti parola prin GETPASS() : ") ;
 if ( strcmp(test,pass) == 0 )
        printf("\nAti nimerit-o !!!") ;
 else
        printf("\nPoate alta data ...") ;
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
/*-------------------------------------------------------------------------------------------*/
```


Scanned with OKEN Scanner

```
/*
        Funcţia citesteParola citeşte de la tastatură o parola 'invizibilă' pentru cei
care ne privesc peste umăr.
        Parametri:
                - pass: tabloul de caractere în care se depune şirul de caractere
                        corespunzător parolei ;
                - lunMax: lungimea maximă permisă;
        Returnează:
                - void;
*/
void citesteParola ( char parola[] , int lunMax )
{
 int i , c ;
 i = 0 ;
 while ( 1 )      {
        c = getch() ;
        if (c==0x00)     // Se ignoră tastele speciale (programul ar putea fi
                         // dezvoltat pentru a trata şi tastele săgeată, Delete, Insert).
                getch() ;
        if (c=='\r')        {        // codul pentru Enter (carriage return)
                parola[i] = '\0' ;
                break ;
                }
        else if (c=='\b')  {        // Backspace
                if (i>0)  {
                        i-- ;
                        cprintf("\b \b") ; // '\b'-spaţiu-'\b'
                        }
                }
        else      {
                if (i==lunMax)  {
                        putch('\a') ;              // Bell alert
                        continue ;
                        }
                parola[i++] = c ;
                putch(0xfe) ;                      // sau orice alt caracter "neutru"
                }
        }
}
```


Scanned with OKEN Scanner

Problema 9.1

Să se scrie şi să se testeze două funcţii pentru crearea unui chenar dreptunghiular avînd conturul dintr-o linie simplă, respectiv dublă.

Se vor folosi următoarele caractere grafice ASCII:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <alloc.h>
#include <string.h>
#include <io.h>
/*
        ATENŢIE : nu se face nici o verificare a coordonatelor transmise în
apelurile funcţiilor (de exemplu pentru stopX < startX, ori coordY = 75)!
*/
/*-------------------------------------------------------------------------------------------*/
/*
        Funcţia linieOrizontala trasează o linie pe orizontală, cu tipul de linie dorit.
        Parametri:
                - startX: coordonata X de start;
                - stopX: coordonata X finală;
                - coordY: coordonata Y la care se trasează linia;
                - tipLinie: cu ce caracter să se traseze linia;
        Returnează:
                - void;
*/
void linieOrizontala ( int startX, int stopX, int coordY, char tipLinie )
{
  int i ;
  gotoxy(startX,coordY) ;
  for (i=startX; i<=stopX; i++)
        putch(tipLinie) ;
}
```


Scanned with OKEN Scanner

```
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia linieVerticala trasează o linie pe verticală, cu tipul de linie dorit.
        Parametri:
                - startY: coordonata Y de start;
                - stopY: coordonata Y finală;
                - coordX: coordonata X la care se trasează linia;
                - tipLinie: cu ce caracter să se traseze linia;
        Returnează:
                - void;
*/
void linieVerticala ( int startY, int stopY, int coordX, char tipLinie )
{
 int i ;
 for (i=startY; i<=stopY; i++)   {
        gotoxy(coordX,i) ;
        putch(tipLinie) ;
        }
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia colt scrie un caracter pe ecran - va fi folosită pentru scrierea
colţurilor dreptunghiurilor.
        Parametri:
                - coordX: coordonata X la care se scrie caracterul;
                - coordY: coordonata Y  "    "   "    "          "
                - tipColt: ce caracter se scrie;
        Returnează:
                - void;
*/
void colt ( int coordX, int coordY, char tipColt )
{
 gotoxy(coordX,coordY) ;
 putch(tipColt) ;
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia dreptunghiDublu desenează pe ecran un dreptunghi avînd chenarul
compus dintr-o linie dublă.
        Parametri:
                - xStinga: coordonate'    olţului stînga-sus;
```


Scanned with OKEN Scanner

- *ySus*
- *xDreapta*: coordonatele colțului dreapta-jos;
- *yJos*

Returnează:

- *void*;

*/

```
void dreptunghiDublu ( int xStinga, int ySus, int xDreapta, int yJos )
{
 colt ( xStinga, ySus, 0xc9) ;
 linieOrizontala ( xStinga+1, xDreapta-1, ySus, 0xcd) ;
 colt ( xDreapta, ySus, 0xbb) ;
 linieVerticala ( ySus+1, yJos-1, xDreapta, 0xba) ;
 colt ( xDreapta, yJos, 0xbc) ;
 linieVerticala ( ySus+1, yJos-1, xStinga, 0xba) ;
 colt ( xStinga, yJos, 0xc8) ;
 linieOrizontala ( xStinga+1, xDreapta-1, yJos, 0xcd) ;
}
/*-------------------------------------------------------------------------------------*/
```

/*

Funcția **dreptunghiSimplu** desenează pe ecran un dreptunghi având chenarul compus dintr-o linie simplă.

Parametri:

- *xStinga*: coordonatele colțului stînga-sus;
- *ySus*
- *xDreapta*: coordonatele colțului dreapta-jos;
- *yJos*

Returnează:

- *void*;

*/

```
void dreptunghiSimplu ( int xStinga, int ySus, int xDreapta, int yJos )
{
 colt ( xStinga, ySus, 0xda) ;
 linieOrizontala ( xStinga+1, xDreapta-1, ySus, 0xc4) ;
 colt ( xDreapta, ySus, 0xbf) ;
 linieVerticala ( ySus+1, yJos-1, xDreapta, 0xb3) ;
 colt ( xDreapta, yJos, 0xd9) ;
 linieVerticala ( ySus+1, yJos-1, xStinga, 0xb3) ;
 colt ( xStinga, yJos, 0xc0) ;
 linieOrizontala ( xStinga+1, xDreapta-1, yJos, 0xc4) ;
}
/*-------------------------------------------------------------------------------------*/
```


Scanned with OKEN Scanner

```
int main ( void )
{
/*
        Testaţi şi fără fixarea valorii pentru _wscroll !
*/
  _wscroll = 0 ;
  clrscr() ;
  dreptunghiSimplu(1,1,80,25) ;
  gotoxy ( 5, 5 );
  printf("\tApasati orice tasta pentru continuare...") ;
  getch() ;
  clrscr() ;
  dreptunghiDublu(3,3,50,20) ;
  gotoxy ( 0, 20 );
  printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
  getch() ;
  return 0 ;
}
```

Problema 9.2

Să se scrie un program care buclează la infinit, afişînd (în video invers) în locurile dorite de pe ecran starea tastelor citită prin apelul **bioskey(2)**.

Pentru afişare se va folosi funcţia **cprintf()** (apelarea se face identic cu **printf()**; va fi detaliată ulterior). Nu uitaţi să "ascundeţi" cursorul !

```
#include <stdio.h>
#include <conio.h>
#include <bios.h>
#include <dos.h>
#define INSERT_ON          0x80
#define CAPS_ON            0x40
#define NUM_LOCK_ON        0x20
#define SCROLL_LOCK_ON     0x10
#define ALT                0x08
#define CTRL               0x04
#define LEFT_SHIFT         0x02
#define RIGHT_SHIFT        0x01
#define ESC                0x1b
int stareInsert = 0 ,
```


Scanned with OKEN Scanner

```
    stareCaps = 0 ,
    stareNum = 0 ,
    stareScroll = 0 ,
    stareAlt = 0 ,
    stareCtrl = 0 ;
int stareLeftShift = 0 ,
    stareRightShift = 0 ;
/*-------------------------------------------------------------------------------------------*/
/*
```

Funcția **controlBreak** preia tratarea întreruperii DOS produse de apăsarea combinației *Ctrl-C* sau *Ctrl-Break*.

Parametri:

- *void*;

Returnează:

- *int*: 0 pentru încheierea programului, altfel 1;

```
*/
int controlBreak ( void )
{
  int c ;
  clrscr() ;
  stareInsert = 0 ;
  stareCaps = 0 ;
  stareNum = 0 ;
  stareScroll = 0 ;
  stareAlt = 0 ;
  stareCtrl = 0 ;
  stareLeftShift = 0 ;
  stareRightShift = 0 ;
  gotoxy(1,23) ;
  cprintf("ESC aborteaza...") ;
  c = getch() ;
  if ( c == ESC )          {
          _setcursortype(_NORMALCURSOR) ;
          return 0 ;
          }
  clrscr() ;
  fflush(stdin) ;
  return 1 ;
}
/*-------------------------------------------------------------------------------------------*/
/*
```


Scanned with OKEN Scanner

Funcţia **normalVideo** schimbă modul de scriere în video normal (alb pe negru, pentru monitoare monocrome).

Parametri:

- *void*;

Returnează:

- *void*;

ATENŢIE: orice astfel de modificare are efect NUMAI asupra scrierilor cu funcţii din grupul **c...** (de ex. **cprintf()**)!

```
*/
void normalVideo ( void )
{
 textcolor(WHITE) ;
 textbackground(BLACK) ;
}
/*-------------------------------------------------------------------------------------*/
/*
```

Funcţia **inversVideo** schimbă modul de scriere în video invers (negru pe alb, pentru monitoare monocrome).

Culoarea WHITE nu este de fapt disponibilă pentru *background*, dar se face automat conversia în,LIGHTGRAY.

Parametri:

- *void*;

Returnează:

- *void*;

```
*/
void inversVideo ( void )
{
 textcolor(BLACK) ;
 textbackground(WHITE) ;
}
/*-------------------------------------------------------------------------------------*/
void main ( void )
{
        int val , test ;
        clrscr() ;
        setcbrk(1) ;
        ctrlbrk(controlBreak) ;
        _setcursortype(_NOCURSOR) ;
        while (1)       {
                val  = bioskey(2) ;
                if ( (val & INSERT_ON) && !stareInsert)       {
```


Scanned with OKEN Scanner

```
            stareInsert = 1 ;
            gotoxy(1,1) ;
            inversVideo() ;
            cprintf("INS") ;
            normalVideo() ;
            }
else if ( !(val & INSERT_ON) && stareInsert)    {
            stareInsert = 0 ;
            gotoxy(1,1) ;
            cprintf("   ") ;              // 3 spații
            }
if ( (val & CAPS_ON) && !stareCaps)   {
            stareCaps = 1 ;
            gotoxy(1,3) ;
            inversVideo() ;
            cprintf("CAPS") ;          // 4 spații
            normalVideo() ;
            }
else if ( !(val & CAPS_ON) && stareCaps)        {
            stareCaps = 0 ;
            gotoxy(1,3) ;
            cprintf("    ") ;
            }
if ( (val & NUM_LOCK_ON) && !stareNum)    {
            stareNum = 1 ;
            gotoxy(1,5) ;
            inversVideo() ;
            cprintf("NUM") ;
            normalVideo() ;
            }
else if ( !(val & NUM_LOCK_ON) && stareNum)         {
            stareNum = 0 ;
            gotoxy(1,5) ;
            cprintf("   ") ;
            }
if ( (val & SCROLL_LOCK_ON) && !stareScroll)        {
            stareScroll = 1 ;
            gotoxy(1,7) ;
            inversVideo() ;
            cprintf("SCROLL") ;
            normalVideo() ;
```


Scanned with OKEN Scanner

```
        }
else if ( !(val & SCROLL_LOCK_ON) && stareScroll)  {
        stareScroll = 0 ;
        gotoxy(1,7) ;
        cprintf("      ") ;
        }
if ( (val & ALT) && !stareAlt)  {
        stareAlt = 1 ;
        gotoxy(1,9) ;
        inversVideo() ;
        cprintf("ALT") ;
        normalVideo() ;
        }
else if ( !(val & ALT) && stareAlt)       {
        stareAlt = 0 ;
        gotoxy(1,9) ;
        cprintf("   ") ;
        }
if ( (val & CTRL) && !stareCtrl)          {
        stareCtrl = 1 ;
        gotoxy(1,11) ;
        inversVideo() ;
        cprintf("CTRL") ;
        normalVideo() ;
        }
else if ( !(val & CTRL) && stareCtrl)     {
        stareCtrl = 0 ;
        gotoxy(1,11) ;
        cprintf("    ") ;
        }
if ( (val & LEFT_SHIFT) && !stareLeftShift)    {
        stareLeftShift = 1 ;
        gotoxy(1,13) ;
        inversVideo() ;
        cprintf("Left-SHIFT") ;
        normalVideo() ;
        }
else if ( !(val & LEFT_SHIFT) && stareLeftShift)       {
        stareLeftShift = 0 ;
        gotoxy(1,13) ;
        cprintf("          ") ;
```


Scanned with OKEN Scanner

```
            }
        if ( (val & RIGHT_SHIFT) && !stareRightShift)        {
                stareRightShift = 1 ;
                gotoxy(1,15) ;
                inversVideo() ;
                cprintf("Right-SHIFT") ;
                normalVideo() ;
                }
        else if ( !(val & RIGHT_SHIFT) && stareRightShift)    {
                stareRightShift = 0 ;
                gotoxy(1,15) ;
                cprintf("          ") ;
                }
        }
}
```

Problema 10.2

Să se scrie un program care citeşte de la tastatură un şir de caractere (reprezentarea unui număr întreg), baza de numeraţie corespunzătoare reprezentării şi o nouă bază de numeraţie, în care trebuie să se convertească numărul şi să se afişeze rezultatul.

Bazele de numeraţie pot fi între 2 şi 16 (inclusiv).

Nu uitaţi să testaţi şi cu "0" pentru numărul de convertit !

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
/*-------------------------------------------------------------------------------------------*/
/*
```

Funcţia **caracterIlegal** determină dacă un caracter este corect folosit într-o anumită bază de numeraţie.

Parametri:

- *chr*: caracterul de verificat;
- *bazaNumeratie*: baza de numeraţie corespunzătoare;

Returnează:

- *int*: 0 dacă este un caracter permis, altfel ;

```
*/
```


Scanned with OKEN Scanner

```
int caracterIlegal ( char chr , int bazaNumeratie )
{
 chr = tolower(chr) ;
 if ( bazaNumeratie < 11 )          {
        if ( chr<'0' || chr>='0'+ bazaNumeratie )
                return 1 ;
        }
else if ( bazaNumeratie > 10 )     {
        if ( chr<'0' || ( chr>'9' && chr<'a' ) || chr >= 'a' + bazaNumeratie - 10 )
                return 1 ;
        }
 return 0 ;
}
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 char sirIntrare[40] ,sirIesire[40], test[40] ;
 int i , baza1 , baza2 , c , semn = 1 ;
 long valoareNumar ;
 clrscr() ;
 printf("\nIntroduceţi reprezentarea numărului întreg : ") ;
 gets ( sirIntrare ) ;
        // Numărul poate fi precedat de white-spaces şi de un eventual semn.
 printf("\n\nBaza de numeraţie corespunzătoare precedentei reprezentări : ") ;
 scanf ( "%d", &baza1 ) ;
 printf("\n\nNoua bază de numeraţie : ") ;
 scanf ( "%d", &baza2 ) ;
 if ( baza1<2 || baza1>16 || baza2<2 || baza2>16 )          {
        printf("\n\aBaze de numeraţie eronate ( în afara 2-16 ) !") ;
        exit(1) ;
        }
                // Se sare peste white-spaces şi se tratează eventualul semn.
for ( i=0; sirIntrare != '\0'; i++ )
        if ( sirIntrare[i] == ' ' || sirIntrare[i] == '\t' )
                ;
        else
                if ( sirIntrare[i] == '-' )
                        semn = -1 ;
        else if ( sirIntrare[i] == '+' )
                        ;
        else
```


Scanned with OKEN Scanner

```
                break ;
    valoareNumar = 0 ;
                        // Se face conversia ASCII -> număr.
    for ( ; sirIntrare[i] != '\0'; i++)  {
        if ( caracterIlegal ( sirIntrare[i], baza1 ) == 1 )    {
                printf("\n\aCaracter ilegal în intrare ! ") ;
                exit(1) ;
                }
        valoareNumar *= baza1 ;
        if ( baza1 < 11 )
                valoareNumar += sirIntrare[i] - '0' ;
        else    {
                if ( sirIntrare[i] <= '9' )
                        valoareNumar += sirIntrare[i] - '0' ;
                else
                        valoareNumar += tolower(sirIntrare[i]) - 'a' + 10 ;
                }
        }
    i = 0 ;
//  Se compară în paralel cu funcția de bibliotecă ltoa() - urmăriți și în HELP -
//semnul '-' se pune numai pentru baza 10 !
    ltoa ( valoareNumar * semn , test, baza2) ;
    if ( valoareNumar == 0 )
        strcpy(sirIesire,"0")       ;
    else    {                       // Se face conversia inversă, număr -> ASCII.
        do      {
                c = (int) valoareNumar % baza2 ;
/*
        Altfel ar apărea avertismentul Conversion may lose significant digits. care
oricum, nu ne deranjează.
*/
                if ( c < 10 )
                        sirIesire[i++] = '0' + c ;
                else
                        sirIesire[i++] = 'a' + c - 10 ; // sau 'A' ( dacă preferăm )
                valoareNumar /= baza2 ;
                }       while ( valoareNumar > 0 ) ;
        if ( semn == -1 )
                sirIesire[i++] = '-' ;
        sirIesire[i] = '\0' ;
        strrev(sirIesire) ;         // Cifrele au fost generate în ordinea inversă.
```


Scanned with OKEN Scanner

```
        }
 printf("\nRezultatul conversiei în noua bază : %s .",sirIesire) ;
 printf("\nComparare cu LTOA()              : %s .",test) ;
 printf("\n\nApăsaţi orice tastă pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 11.3

Scrieţi un program care citeşte linii text de la tastatură, le depune în zone de memorie alocate dinamic, după care le afişează în ordinea crescătoare / descrescătoare a lungimilor lor.

```
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <alloc.h>
/*-------------------------------------------------------------------------------*/
int main ( void )
{
  char *adreseLinii[20] , linieText[80] , *temp ;
  int i=0 , numarLinii , inv ;
  clrscr() ;
  printf("\nIntroduceti liniile ( incheiere cu o linie vida ) :\n") ;
  while ( gets(linieText),.strlen(linieText) )        {   // atenţie la operatorul virgulă
        adreseLinii[i] = (char *) malloc ( strlen(linieText)+1 ) ;
                                  // strlen() nu contorizează şi '\0' final.
        if ( adreseLinii[i] == 0 ) {
              printf("\n\aEroare alocare memorie !") ;
              exit(1) ;
              }
        strcpy ( adreseLinii[i++], linieText ) ;
        if ( i > 19 )                        // S-a atins maximul alocat.
              break ;
        }
// ATENŢIE : eroarea tipică ar fi aici să se scrie direct adreseLinii[i] = linieText ;
//           sau chiar să se "sară" peste alocarea memoriei şi să se lucreze cu
//           pointeri neinitializaţi !
```


Scanned with OKEN Scanner

```
/*
        Ceea ce se sortează este tabloul de adrese ale liniilor text! Nu se face nici o
rescriere a liniilor dintr-o zonă de memorie în alta (pe lîngă timpul consumat, ar fi
necesare verificări ale memoriei alocate pentru adresele cu care se operează,
eventual măriri ale spațiilor alocate - prin realloc()).
*/
 numarLinii = i ;                          //numărul de linii efectiv introduse
 do     {
        inv = 0 ;
        for (i=0; i<numarLinii-1; i++)
                if ( strlen ( adreseLinii[i] ) > strlen ( adreseLinii[i+1] ) )   {
                        temp = adreseLinii[i] ;
                        adreseLinii[i] = adreseLinii[i+1] ;
                        adreseLinii[i+1] = temp ;
                        inv = 1 ;
                        }
        } while ( inv ) ;
 printf("\nLiniile sortate crescător după lungime :\n\n") ;
 for (i=0; i<numarLinii; i++)
        printf("%s\n",adreseLinii[i]) ;
 printf("\nApăsați o tastă pentru afişarea în ordinea descrescătoare :\n\n") ;
 getch() ;
 for (i=numarLinii-1; i>=0; i--)
        printf("%s\n",adreseLinii[i]) ;
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 12.2

Să se testeze expandarea argumentelor din linia de comandă ce conțin *wildcards* ('*' sau '?').

Parametrii liniei de comandă se pot indica fie de la prompt-ul DOS, fie din meniul

Run|Arguments... .

Dacă se doresc *wildcards*, NU se operează în fişierele sursă, trebuie creat un proiect în care să se includă şi fişierul ***WILDARGS.OBJ*** (din directorul ***\TC\LIB***)!

```
#include <stdio.h>
```


Scanned with OKEN Scanner

```
#include <conio.h>
/*-------------------------------------------------------------------------------*/
int main ( int argc , char *argv[] , char *env[] )
{
 int i ;
 clrscr() ;
 printf("ARGC = %d\n\n",argc) ;
 for (i=0; argv[i]!=NULL; i++)
        printf("argv[%d] : %s\n",i,argv[i]) ;
 printf("\n\tApasati orice tasta pentru continuare...") ;
 getch() ;
 for (i=0; env[i]!=NULL; i++)
        printf("env[%d] : %s\n",i,env[i]) ;
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 12.4

Scrieţi un program care afişează variabilele de mediu DOS, modifică (prin **putenv()**) calea de căutare a comenzii şi reafişează variabilele de mediu.

Pentru comparare, se vor lansa din programul C comenzile DOS **SET** (fără parametri; va produce afişarea variabilelor de mediu) şi **PATH** (pentru afişarea căii de căutare a comenzii).

Nu omiteţi testarea succesului funcţiei **system()**!

ATENŢIE : dacă se foloseşte modelul de memorie *tiny* sau *small*, NU se va putea lansa echivalentul unei linii de comandă DOS prin execuţia din IDE (se produce eroarea sistem *Not enough memory*)!

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
/*-------------------------------------------------------------------------------*/
int main ( void )
{
  clrscr() ;
  printf("Afisare variabile de mediu DOS :\n") ;
  if ( system("SET")!=0 )
```


Scanned with OKEN Scanner

```
        perror("\nLansare SYSTEM ") ;
 if ( putenv("PATH=.") )
        perror("PUTENV ") ;
 printf("\nCalea \"citită\" cu getenv() : %s",getenv("PATH")) ;
 printf("\nCalea afisat... prin PATH     : ") ;
 if ( system("PATH")!=0 )
        perror("\nLansare PATH ") ;
 printf("\nReafisare variabile de mediu DOS :\n") ;
 if ( system("SET")!=0 )
        perror("\nLansare SYSTEM ") ;
 printf("\n\nApăsaţi orice tastă pentru terminare ...") ;
 getch() ;
 return 0 ;
}
```

Observaţie: Calea (*PATH*) va fi refăcută automat la cea anterioară odată cu terminarea programului.

Problema 13.2

Scrieţi un program care citeşte de la tastatură maximum 20 de linii a cîte maximum 35 de caractere, după care împarte ecranul în două ferestre (cu chenar) pe verticală şi scrie în stînga liniile citite, iar în dreapta aceleaşi linii după sortare (lexicografică). Un parametru opţional în linia de comandă va face ca rezultatul sortării să fie afişat în video invers.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <alloc.h>
#include <string.h>
#include <io.h>
//  Se refolosesc funcţiile prezentate la problema 9.1 (nu se mai prezintă încă o dată
// codul corespunzător).
/*-------------------------------------------------------------------------------------*/
int main ( int argc , char *argv[] )
/*
        Se alege varianta de apelare      13_2 [/I], fără a se semnala eventualii
parametri incorecţi.
```


Scanned with OKEN Scanner

```
*/
{
 char *adreseLinii[20] , linie[40] , *temp ;
 int i , inv , numarLinii , videoInvers ;
 clrscr() ;
 videoInvers = 0 ;
 if ( argc>1 && !stricmp(argv[1],"/I") )
                videoInvers = 1 ;
 printf("\nIntroduceti liniile :\n") ;
 linie[0] = 35 ;
 for ( i=0; i<20; i++)    {
        cgets(linie) ;
        if ( linie[1]==0 )          break ;
        printf("\n") ;                          // obligatoriu !
        adreseLinii[i] = (char *) malloc ( linie[1] + 1 ) ;
        if ( adreseLinii[i] == NULL )    {
                printf("\n\aEroare alocare memorie !") ;
                exit(1) ;
                }
        strcpy(adreseLinii[i],&linie[2]) ;
        }
 numarLinii = i ;
 clrscr() ;
 dreptunghiDublu(1,1,40,22) ;
 window(2,2,39,21) ;
 for ( i=0; i<numarLinii; i++)
        cprintf("%s\n\r",adreseLinii[i]) ;
                // cprintf() nu converteste '\n' în perechea '\n'-'\r' !
 window(1,1,80,25) ;
 dreptunghiDublu(41,1,80,22) ;
 window(42,2,79,21) ;
 if ( videoInvers == 1 )  {
        textcolor(BLACK) ;
        textbackground(WHITE) ; // este convertit automat în LIGHTGRAY !
        }
 clrscr() ;
 do     {
        inv = 0 ;
        for ( i=0; i<numarLinii-1; i++)
                if ( strcmp(adreseLinii[i],adreseLinii[i+1])>0 )    {
                        temp = adreseLinii[i] ;
```


Scanned with OKEN Scanner

```
                        adreseLinii[i] = adreseLinii[i+1] ;
                        adreseLinii[i+1] = temp ;
                        inv = 1 ;
                        }
        } while(inv) ;
for ( i=0; i<numarLinii; i++)
        cprintf("%s\n\r",adreseLinii[i]) ;
window(1,1,80,25) ;
gotoxy ( 20 , 24 ) ;
printf("Apasati orice tasta pentru terminare...") ;
getch() ;
if ( videoInvers == 1 )  {
        textcolor(WHITE) ;
        textbackground(BLACK) ;
        }
clrscr() ;
return 0 ;
}
```

Problema 13.3

Scrieți o funcție proprie care să emuleze funcția (de bibliotecă) **_strerror()**.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
/*-------------------------------------------------------------------------------------------*/
/*
        Funcţia StrError este echivalentul pentru funcția de bibliotecă _strerror() :
afişează un mesaj de eroare personalizat.
        Parametri:
                - mesaj: mesajul utilizator pentru eroarea curentă;
        Returnează:
                - char *: mesajul de eroare complet;
*/
char *StrError ( char *mesaj )
{
 static char buf[128] ;
 sprintf ( buf,"%s: %s\n", mesaj, sys_errlist[errno] ) ;
```


Scanned with OKEN Scanner

```
 return buf ;
}
/*-------------------------------------------------------------------------------*/
int main ( void )
{
 FILE *f ;
 clrscr() ;
 f = fopen("EXISTA.NU!","rb") ;          // sau orice nume de fişier inexistent
 printf("\nBiblioteca TC   : \n%s",_strerror("Cauza erorii ")) ;
 printf("\n\nFunctia proprie : \n%s",StrError("Cauza erorii ")) ;
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 13.4

Să se scrie şi să se testeze o funcţie pentru afişarea unor linii de caractere într-o fereastră (fiind vizibile simultan maximum 5). Linia curentă va fi în video invers; se trece de la o linie la alta cu tastele *Up* sau *Dn* (eventual şi *PageUp* - *PageDn*).

Se revine din funcţie tastînd *Esc* (0x1a - se va returna -1) sau *Enter* (se va returna numărul de ordine al liniei curente).

Octeţii generaţi de tastele necesare (al doilea după 0x00) sînt:

Dn	0x50	*PageDn*	0x51
Up	0x48	*PageUp*	0x49
Right	0x4a	*Left*	0x4b

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <alloc.h>
#include <string.h>
#include <io.h>
#define ESC         0x1b
#define ENTER       '\r'
#define UP          0x48
#define DN          0x50
```


Scanned with OKEN Scanner

```
#define PgUp            0x49
#define PgDn            0x51
//  Se refolosesc funcţiile prezentate la problema 9.1 (nu se mai prezintă încă o dată
// codul corespunzător).
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia sageataSus tratează cazul apăsării tastei Up.
        Parametri:
            - adrL: adresele liniilor text ce se afişează;
            - nrLin: numărul de linii;
            - &nrLinCr: adresa pentru variabila ce pastrează numărul de ordine
                        al liniei curente din fereastră (1 ÷ 5);
            - &prLin: adresa pentru variabila ce pastrează numărul de ordine (în
                      fişier!) al primei linii afişate în fereastră;
            - &ultLin: adresa pentru variabila ce pastrează numărul de ordine (în
                       fişier!) al ultimei linii afişate în fereastră;
        Returnează:
                - void;
*/
void sageataSus ( char *adrL[] , int nrLin , int *nrLinCr , int *prLin , int *ultLin )
{
 int i ;
 if ( *nrLinCr==1 && *prLin==0 )       // sîntem pe prima linie din total
        return ;
 if ( (*nrLinCr) > 1 )    {
            // Se reafişează linia curentă în video normal, după care se trece pe linia
            // precedentă, afişînd-o în video invers.
        cprintf ( "%s\r", adrL[*prLin + *nrLinCr - 1] ) ;
        (*nrLinCr)-- ;
        gotoxy ( 1, *nrLinCr ) ;
        textattr(0x70) ;
        cprintf ( "%s\r", adrL[*prLin + *nrLinCr - 1] ) ;
        textattr(0x0f) ;
        }
 else   {       // Se 'pierde' ultima linie din afişaj, făcînd defilare la linia anterioară.
        clrscr() ;
        (*prLin)-- ;
        for ( i=0; i<nrLin && i<5; i++)  {
                cprintf ( "%s", adrL[*prLin + i] ) ;
                if ( i<4 )
                        cprintf("\n\r") ;
```


Scanned with OKEN Scanner

```
                else
                        cprintf("\r") ;
                }
        gotoxy(1,1) ;
        textattr(0x70) ;                    // altă variantă pentru video invers
        cprintf ( "%s\r", adrL[*prLin] ) ;
        textattr(0x0f) ;                    // altă variantă pentru video normal
        (*ultLin)-- ;
        }
}
/*-------------------------------------------------------------------------------------------*/
/*
        Funcția sageataJos tratează cazul apăsării tastei Dn.
        Parametri:
            - adrL: adresele liniilor text ce se afişează;
            - nrLin: numărul de linii;
            - &nrLinCr: adresa pentru variabila ce pastrează numărul de ordine
                        al liniei curente din fereastră (1 ÷ 5);
            - &prLin: adresa pentru variabila ce pastrează numărul de ordine (în
                        fişier!) al primei linii afişate în fereastră;
            - &ultLin: adresa pentru variabila ce pastrează numărul de ordine (în
                        fişier!) al ultimei linii afişate în fereastră;
        Returnează:
                - void;
*/
void sageataJos ( char *adrL[] , int nrLin , int *nrLinCr , int *prLin , int *ultLin )
{
 int i ;
 if ( *ultLin == nrLin-1 && ( *nrLinCr == 5 || *nrLinCr == nrLin ) )
                return ;                    // tastă inoperantă ( logic )
 if ( (*nrLinCr) <= 4 )  {
        cprintf ( "%s\r", adrL[*prLin + *nrLinCr - 1 ] ) ;
        (*nrLinCr)++ ;
        gotoxy ( 1, *nrLinCr ) ;
        textattr(0x70) ;
        cprintf( "%s\r", adrL[*prLin + *nrLinCr - 1 ] ) ;
        textattr(0x0f) ;
        }
else    {
        clrscr() ;
        (*prLin)++ ;
```


Scanned with OKEN Scanner

```
        for ( i=0; i<nrLin && i<5; i++)  {
                cprintf ( "%s", adrL[*prLin + i] ) ;
                if ( i<4 )
                        cprintf("\n\r") ;
                else
                        cprintf("\r") ;
                }
        gotoxy(1,i) ;
        textattr(0x70) ;
        cprintf ( "%s\r", adrL[*prLin + *nrLinCr - 1 ] ) ;
        textattr(0x0f) ;
        (*ultLin)++ ;
        }
}
/*-------------------------------------------------------------------------------------------*/
/*
```

Funcţia **afisareTablou** face afişarea liniilor conform enunţului problemei.
Parametri:
- *adrL*: adresele liniilor text ce se afişează;
- *nrLin*: numărul de linii;

Returnează:
- *int*: numărul de ordine al liniei selectate prin *Enter*, respectiv -1 dacă s-a tastat *ESC*;

```
*/
int afisareTablou ( char *adrL[] , int nrLin )
{
 int i , k , c , nrLinieCurenta , primaLinie , ultimaLinie ;
 if ( nrLin<2 )
        return 0 ;
 _setcursortype(_NOCURSOR) ;
 dreptunghiDublu(10,10,47,16) ;
 window(11,11,46,15) ;
 clrscr() ;
 for ( i=0; i<nrLin && i<5; i++)          {
        cprintf ( "%s", adrL[i] ) ;
        if ( i<4 )
                cprintf("\n\r") ;
        else
                cprintf("\r") ;
        }
 gotoxy(1,1) ;
```


Scanned with OKEN Scanner

```
textattr(0x70) ;          // video invers
cprintf ( "%s\r", adrL[0] ) ;
textattr(0x0f) ;          // video normal
nrLinieCurenta = 1 ;      // numărul liniei curente în fereastră - 1÷5
primaLinie = 0 ;          // numărul primei linii afişate în fereastră - 0÷nrLin-1
ultimaLinie = i-1 ;       // numărul ultimei linii afişate în fereastră - 0÷nrLin-1
while ( 1 )       {
  c = getch() ;
  if ( c==ESC )
       return(-1) ;
  if ( c==ENTER )
       return ( primaLinie + nrLinieCurenta - 1 ) ;
  if ( c==0x00 )          {
       c = getch() ;
       if ( c==UP )
         sageataSus ( adrL , nrLin , &nrLinieCurenta ,
                      &primaLinie , &ultimaLinie ) ;
       else if ( c==DN )
         sageataJos ( adrL , nrLin , &nrLinieCurenta ,
                      &primaLinie , &ultimaLinie ) ;
       else if ( c==PgUp )        // se echivalează cu 5 * Up
         for ( k=0; k<5; k++ )
           sageataSus ( adrL , nrLin , &nrLinieCurenta ,
                        &primaLinie , &ultimaLinie ) ;
       else if ( c==PgDn )     // se echivalează cu 5 * Dn
         for ( k=0; k<5; k++ )
           sageataJos ( adrL , nrLin , &nrLinieCurenta ,
                        &primaLinie , &ultimaLinie ) ;
       }
  }
}
/*-------------------------------------------------------------------------------*/
int main ( void )
{
 char *adreseLinii[20] , linie[40] ;
 int i , nrLinii , k ;
 clrscr() ;
 printf("\nIntroduceti liniile ( max. 20 ) :\n") ;
 linie[0] = 36 ;
 for ( i=0; i<20; i++)     {
       cgets(linie) ;
```


Scanned with OKEN Scanner

```
        if ( linie[1]==0 )          break ;
        printf("\n") ;                          // obligatoriu !
           // Se completează cu spaţii pînă la lungimea maximă a fiecărei linii .
        for ( k = 2 + linie[1]; k<37; k++)
                linie[k] = ' ' ;
        linie[37] = '\0' ;
        adreseLinii[i] = (char *)malloc(36) ;
        if ( adreseLinii[i] == NULL )     {
                printf("\n\aEroare alocare memorie !") ;
                exit(1) ;
                }
        strcpy ( adreseLinii[i], &linie[2] ) ;
        }
  nrLinii = i ;
  clrscr() ;
  i = afisareTablou ( adreseLinii , nrLinii ) ;
  window(1,1,80,25) ;
  clrscr() ;
  _setcursortype(_NORMALCURSOR) ;
  if ( i == -1 )
    printf("\nS-a revenit tastînd ESC .") ;
  else
    printf("\n\nS-a revenit cu ENTER de pe linia a %d-a : \n%s\n",
                                        i+1,adreseLinii[i]) ;
  printf("\n\nApăsaţi orice tastă pentru terminare..." ) ;
  getch() ;
  return 0 ;
}
```

Problema 14.1

Scrieţi şi testaţi o funcţie ***char *elimCaracterDinSir (char *a, const char *b)*** care elimină din şirul de caractere ***a*** toate caracterele care fac parte şi din şirul de caractere ***b***.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
/*-------------------------------------------------------------------------------*/
/*
```


Scanned with OKEN Scanner

Funcția **elimCaracterDinSir** elimină toate aparițiile unui caracter dintr-un șir de caractere.

Parametri:

- *s*: șirul din care se face eliminarea;
- *ch*: caracterul ce se elimină;

Returnează:

- *char* *: șirul rezultat în urma operațiunii;

```
*/
char * elimCaracterDinSir ( char *s , const char ch )
{
 int i, j ;
 char *retVal = s ;
 for ( i=0; s[i]!='\0'; )
        if ( s[i] == ch )  {
                for ( j=i; s[j]!='\0'; j++)
                        s[j] = s[j+1] ;
//      Se face translarea cu o poziție la stînga a tuturor caracterelor pornind de la
// următorul indice.
                }
        else
                i++ ;
 return retVal ;
}
/*-------------------------------------------------------------------------------------*/
/*
```

Funcția **elim** elimină toate aparițiile caracterelor ce fac parte dintr-un șir de caractere din alt șir.

Parametri:

- *a*: șirul din care se face eliminarea;
- *b*: șirul cu caracterele ce se elimină;

Returnează:

- *char* *: șirul rezultat în urma operațiunii;

```
*/
char * elim ( char *a , const char *b )
{
  char *retVal = a ;
  int i ;
  for ( i=0; b[i] != '\0'; i++ )
        elimCaracterDinSir ( a, b[i] ) ;
  return retVal ;
}
```


Scanned with OKEN Scanner

```
/*-------------------------------------------------------------------------------*/
int main ( void )
{
 char sir[100] , sir2[100], carElim ;
 clrscr() ;
 printf("Sirul de referinţă  : ") ;
 gets(sir) ;
 printf("\n\nCaracterul de eliminat : ") ;
 scanf ( " %c", &carElim ) ;                    // NU uitati spaţiul înainte de %c !
 elimCaracterDinSir ( sir, carElim ) ;
 printf("\n\nIn urma eliminării a rămas : %s",sir) ;
 printf("\nSirul avînd rolul \"b\" : ");
 gets(sir2) ;
 elim(sir,sir2) ;
 printf("\n\nIn urma eliminării a rămas : %s",sir) ;
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 14.2

Scrieţi şi testaţi o funcţie ***int last (const char *a, const char *b)*** care să returneze indicele ultimei apariţii a şirului de caractere ***b*** ca subşir al lui ***a*** (-1 cînd nu este inclus în ***a***).

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
/*-------------------------------------------------------------------------------*/
/*
        Funcţia ultimaAp1 caută ultima apariţie a unui şir de caractere în altul - se
bazează pe funcţia de bibliotecă strstr().
        Parametri:
                - a: şirul în care se caută;
                - b: şirul ce se caută;
        Returnează:
                - int: indicele corespunzator ultimei apariţii
                    (-1 dacă nu există nici una);
*/
```


Scanned with OKEN Scanner

```
int ultimaAp1 ( const char *a , const char *b )
{
 int i ;
/*
        Se porneşte de la sfîrşitul şirului a către început, deci prima apariţie
detectată este de fapt ultima în sensul enunţului problemei.
*/
 for ( i=strlen(a)-1; i>=0; i-- )
        if ( strstr ( &a[i], b ) != NULL )
                return i ;
 return -1 ;
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia ultimaAp2 caută ultima apariţie a unui şir de caractere în altul - se
bazează pe funcţia de bibliotecă strncmp().
        Parametri:
                - a: şirul în care se caută;
                - b: şirul ce se caută;
        Returnează:
                - int: indicele corespunzător ultimei apariţii
                    (-1 dacă nu există nici una);
*/
int ultimaAp2 ( const char *a , const char *b )
{
 int i , j=-1 ;
 for ( i=0; a[i]!='\0'; i++ )
        if ( !strncmp ( &a[i], b, strlen(b) ) )
                j = i ;
 return j ;
}
/*-------------------------------------------------------------------------------------*/
int main ( void )
{
 char sir1[50] , sir2[50] ;
 int pozitia ;
 clrscr() ;
 printf("Şirul de referinţă  : ") ;
 gets(sir1) ;
 printf("\n\nŞirul de căutat : ") ;
 gets(sir2) ;
```


Scanned with OKEN Scanner

```
pozitia = ultimaAp1 ( sir1 , sir2 ) ;
printf("\n\n\nCu prima functie ( STRSTR() ) :\n") ;
if ( pozitia==-1 )
        printf("Al doilea sir NU apare în primul !") ;
else
        printf("Ultima aparitie este la indicele %d .",pozitia) ;
printf("\n\n\nCu a doua functie ( STRNCMP() ) :\n") ;
pozitia = ultimaAp2 ( sir1 , sir2 ) ;
if ( pozitia==-1 )
        printf("Al doilea sir NU apare în primul !") ;
else
        printf("Ultima aparitie este la indicele %d .",pozitia) ;
printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
getch() ;
return 0 ;
}
```

Problema 14.3

Reluați programul de sortare a unor linii text introduse de la tastatură și memorate în zone de memorie alocate dinamic **(11.3)**, adăugînd și sortarea folosind funcția de bibliotecă **qsort()**; încercați măsurarea timpului consumat în ambele variante **(clock())**, făcînd afișarea rezultatelor în paralel.

Introduceți doi parametri opționali în linia de comandă : unul care să impună ordonarea descrescătoare (implicit: crescătoare) și altul care să impună sortarea numerică (liniile text să fie interpretate ca reprezentarea ASCII a unor numere întregi).

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <alloc.h>
#include <string.h>
#include <io.h>
#include <time.h>
//  Se refolosesc funcțiile prezentate la problema 9.1 (nu se mai prezintă încă o dată
// codul corespunzător).
/*-------------------------------------------------------------------------------*/
/*
```


Scanned with OKEN Scanner

Funcția **compLexCresc** se va folosi la sortarea lexicografic crescătoare prin **qsort()**.

Parametri:

- *a*: adresele pointerilor pe char reprezentînd adresele liniilor text respective
- *b*

Returnează:

- *int*: rezultatul comparării;

```
*/
int compLexCresc ( char **a , char **b )
{
    return ( strcmp(*a,*b) ) ;
}
/*-------------------------------------------------------------------------------*/
/*
```

Funcția **compLexDescresc** se va folosi la sortarea lexicografic descrescătoare prin **qsort()**.

Parametri:

- *a*: adresele pointerilor pe char reprezentînd adresele liniilor text respective
- *b*

Returnează:

- *int*: rezultatul comparării;

```
*/
int compLexDescresc ( char **a , char **b )
{
    return ( strcmp(*b,*a) ) ;
}
/*-------------------------------------------------------------------------------*/
/*
```

Funcția **compNumCresc** se va folosi la sortarea numeric crescătoare prin **qsort()**.

Parametri:

- *a*: adresele pointerilor pe char reprezentînd adresele liniilor text respective
- *b*

Returnează:

- *int*: rezultatul comparării;

```
*/
int compNumCresc ( char **a , char **b )
{
```


Scanned with OKEN Scanner

```
        return ( atoi(*a) - atoi(*b) ) ;
}
/*-------------------------------------------------------------------------------*/
/*
        Funcţia compNumDescresc se va folosi la sortarea numeric descrescătoare
prin qsort().
        Parametri:
                - a: adresele pointerilor pe char reprezentînd adresele liniilor text
                     respective
                - b
        Returnează:
                - int: rezultatul comparării;
*/
int compNumDescresc ( char **a , char **b )
{
        return ( atoi(*b) - atoi(*a) ) ;
}
/*-------------------------------------------------------------------------------*/
int main ( int argc , char *argv[] )
/*
        Apelul se face de forma : 14_3 [/N] [/D]
*/
{
 char *adrLin1[20] , *adrLin2[20] , linie[40] , *temp ;
 int i , inv , nrLinii ;
 clock_t a1 , a2 , b1 , b2 ;
 int cresc = 1,              // va fi 0 la ordonarea descrescătoare
     numer = 0 ;             // 0 - sortare lexicografică, 1 - numerică
 clrscr() ;
 if ( argc>1 )    {
        if ( !stricmp(argv[1],"/D") )
                cresc = 0 ;
        else if ( !stricmp(argv[1],"/N") )
                numer = 1 ;
        }
if ( argc>2 )     {
        if ( !stricmp(argv[2],"/D") )
                cresc = 0 ;
        else if ( !stricmp(argv[2],"/N") )
                numer = 1 ;
        }
```


Scanned with OKEN Scanner

```
printf("\nIntroduceti liniile :\n") ;
linie[0] = 35 ;
for ( i=0; i<20; i++)     {
        cgets(linie) ;
        if ( linie[1]==0 )          break ;
        printf("\n") ;
        adrLin1[i] = (char *)malloc(linie[1]+1) ;
        if ( adrLin1[i]==NULL )          {
                printf("\n\aEroare alocare memorie !") ;
                exit(1) ;
                }
        adrLin2[i] = adrLin1[i] ;
        strcpy ( adrLin1[i], &linie[2] ) ;
        }
nrLinii = i ;
clrscr() ;
dreptunghiDublu(1,1,40,22) ;
window(2,2,39,21) ;
a1 = clock() ;
do      {
  inv = 0 ;
  for ( i=0; i<nrLinii-1; i++)
    if ( cresc ? ( numer ? ( atoi(adrLin1[i]) > atoi(adrLin1[i+1]) ) :
                           ( strcmp(adrLin1[i],adrLin1[i+1]) > 0 ) ) :
               ( numer ? ( atoi(adrLin1[i]) < atoi(adrLin1[i+1]) ) :
                           ( strcmp(adrLin1[i],adrLin1[i+1]) < 0 ) ) )  {
        temp = adrLin1[i] ;
        adrLin1[i] = adrLin1[i+1] ;
        adrLin1[i+1] = temp ;
        inv = 1 ;
        }
    }  while(inv) ;
 a2 = clock() ;
 for ( i=0; i<nrLinii; i++)
        cprintf("%s\n\r",adrLin1[i]) ;
 window(1,1,80,25) ;
 dreptunghiDublu(41,1,80,22) ;
 window(42,2,79,21) ;
 b1 = clock() ;
 if ( cresc && !numer )
        qsort ( adrLin2, nrLinii, sizeof(char *),
```


Scanned with OKEN Scanner

```
                (int(*)(const void *,const void *))compLexCresc) ;
else if ( !cresc && !numer )
        qsort ( adrLin2, nrLinii, sizeof(char *),
                (int(*)(const void *,const void *))compLexDescresc) ;
else if ( cresc && numer )
        qsort ( adrLin2, nrLinii, sizeof(char *),
                (int(*)(const void *,const void *))compNumCresc) ;
else if ( !cresc && numer )
        qsort ( adrLin2, nrLinii, sizeof(char *),
                (int(*)(const void *,const void *))compNumDescresc) ;
b2 = clock() ;
for ( i=0; i<nrLinii; i++)
        cprintf("%s\n\r",adrLin2[i]) ;
getch() ;
window(1,1,80,25) ;
clrscr() ;
printf("\nTimp \"propriu\" .: %f\n",(a2-a1)/CLK_TCK) ;
printf("\nTimp qsort     : %f\n",(b2-b1)/CLK_TCK) ;
//  Atenţie la timpii "nuli" - eventual se repetă operaţiunea de ( să zicem ) 100 de ori
// şi se împarte apoi timpul global ( deşi rezultatele nu mai sînt foarte revelatoare ) la
// numărul de repetări.
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 15.3

Scrieţi o funcţie de căutare (prin metoda înjumătăţirii intervalului - [2]) a rădăcinii unei funcţii (indicată prin nume) într-un interval (limitele intervalului şi precizia cu care se lucrează vor fi tot parametri - se vor testa parametrii; se presupune că funcţia are cel mult o rădăcină în intervalul indicat - [2]).

Testaţi programul atît cu o funcţie proprie, cît şi cu unele din cele incluse în **MATH.H**.

Creaţi o funcţie cu prototipul :

```
int injumInterval ( double ls, double ld, double eps,
                    double *rad, double (*func)() );
```

- *ls* şi *ld* sînt limitele intervalului de căutare ;

Scanned with OKEN Scanner

- *eps* este precizia cu care se caută rădăcina;
- funcţia să returneze 0 pentru succes, respectiv 1 pentru eşec (ls≥ld sau func(ls)*func(ld)>0);
- în varianta pre-ANSI, apelul funcţiei indicată prin pointer s-ar face în forma:

(*func)(x)

- standardul ANSI permite şi direct: f(x);
- nu uitaţi posibilitatea ca rădăcina să fie într-una din cele două limite ale intervalului sau în vreun mijloc;

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
/*-------------------------------------------------------------------------------*/
/*
        Funcţia injumInterval caută rădăcina unei funcţii într-un interval indicat,
cu precizia dorită.
        Parametri:
                - limSt: limita stîngă a intervalului;
                - limDr: limita dreaptă a intervalului;
                - eps: precizia cu care se determină rădăcina;
                - rad: adresa variabilei ce va căpăta valoarea rădăcinii;
                - f: numele functiei a cărei rădăcină se caută
                    (trebuie să existe ca un modul compilat!)
        Returnează:
                - int: 0 dacă s-a putut determina corect o rădăcină, altfel 1;
*/
int injumInterval ( double limSt , double limDr , double eps ,
                    double *rad, double (*f)() )
{
/*
        Vor apărea 6 avertismente de forma
                Call to function with no prototype in function ...
        Se pot elimina (deşi s-ar produce eventual eroare la alte funcţii) prin :
                                        ..., double (*f)(double) )
*/
 double mijlocInt ;
 if ( limSt >= limDr || eps <= 0 )
        return 1 ;
 if ( f(limSt) * f(limDr) >= 0 )
        return 1 ;
```


Scanned with OKEN Scanner

```
 while ( limDr - limSt > eps )     {
        mijlocInt = ( limSt + limDr ) / 2 ;
        if ( f(mijlocInt) == 0 )  {
                *rad = mijlocInt;
                return 0 ;
                }
        if ( f(limSt) * f(mijlocInt) > 0 )
                limSt = mijlocInt ;
        else
                limDr = mijlocInt ;
        }
 *rad = ( limSt + limDr ) / 2 ;    // de fapt orice valoare din intervalul rămas
 return 0 ;
}
/*-------------------------------------------------------------------------------*/
int main ( void )
{
 double limStinga , limDreapta , precizia , valRadacina ;
 clrscr() ;
 printf("\n\nSe cauta o radacina a functiei SINUS :\n\n");
 printf("Limita stinga a intervalului : ") ;
 scanf("%lf",&limStinga) ;
 printf("Limita dreapta a intervalului : ") ;
 scanf("%lf",&limDreapta) ;
 printf("Precizia : ") ;
 scanf("%lf",&precizia) ;
 if ( injumInterval ( limStinga, limDreapta, precizia, &valRadacina, sin) )
        printf("\n\aEroare !\n") ;                    // s-ar putea rafina
 else
        printf("Radacina este %lf .",valRadacina) ;
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 16.2

Scrieți un program care determină și afișează toate datele de *Vineri 13* de la începutul anului curent (introdus ca referință) pînă într-un an indicat (eventual) ca argument în linia de comandă (implicit 2010).

Scanned with OKEN Scanner

```
#include <stdio.h>
#include <conio.h>
int lungimeLuna[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 } ;
char *numeLuna[12] = { "ianuarie" , "februarie" , "martie" , "aprilie" ,
                       "mai" , "iunie" , "iulie" , "august" , "septembrie" ,
                       "octombrie" , "noiembrie" , "decembrie" } ;
/*-------------------------------------------------------------------------------------------*/
int main(int argc,char *argv[])
{
  int anCurent , anFinal , lunaCurenta ;
  long nrZile ;  // contorizează numarul de zile scurse de la referință (1 Ian. 1996)
                 // pînă la începutul lunii curente
  nrZile = 0 ;
  clrscr() ;
  if ( argc > 1 )  {
        if ( sscanf ( argv[1], "%d", &anFinal) != 1 )
              anFinal = 2000 ;
        if ( anFinal < 1997 )
              anFinal = 2000 ;
        }
  else
        anFinal = 2000 ;
  for (anCurent=1996 ; anCurent<=anFinal ; anCurent++)
        for (lunaCurenta=0; lunaCurenta<12; lunaCurenta++)     {
           if ( (nrZile+13) % 7 == 5 )              // 1996: 1 Ianuarie este luni
              printf("%d : vineri 13 %s\n",anCurent, numeLuna[lunaCurenta] ) ;
           nrZile += lungimeLuna[lunaCurenta] ;
           if (lunaCurenta==1)  // se adună 1 pentru februarie şi an bisect
                nrZile += anCurent%4==0 && anCurent%100!=0 ||
                                                anCurent%400==0 ;
           }
  printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
  getch() ;
  return 0 ;
}
```

Problema 16.3

Scanned with OKEN Scanner

Scrieți un program care determină și afișează în ce zi a săptămînii "cade" 1 Ianuarie din anul curent (al cărui început este referință) pînă într-un an indicat prin dialog (implicit: 2010).

```
#include <stdio.h>
#include <conio.h>
char *numeZile[] = { "luni" , "marti" , "miercuri" , "joi" ,
                     "vineri" , "simbata" , "duminica" } ;
/*-------------------------------------------------------------------------------------*/
int main ( void )
{
  int i , anFinal , anBisect ;
  long nrZile ;
  clrscr() ;
  printf("An limita : ") ;
  if ( scanf("%d",&anFinal) != 1 || anFinal < 1997 )
        anFinal = 2010 ;
  nrZile = 0 ;                              // 1 Ian. 1996 : luni
  for ( i=1996 ; i<=anFinal ; i++ )         {
        printf(" 1 Ian. %d : %s\n",i,numeZile[nrZile%7]) ;
        anBisect = i%4==0 && i%100!=0 || i%400==0 ;
        nrZile += 365 + anBisect ;
        }
  printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
  getch() ;
  return 0 ;
}
```

Problema 16.5

Scrieți și testați o funcție care să realizeze înmulțirea a două tablouri bidimensionale (de numere întregi) ale căror dimensiuni se indică prin dialog.

```
#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <conio.h>
/*-------------------------------------------------------------------------------------*/
/*
```

Funcția **inmultMatr** realizează înmulțirea a două matrice.

Scanned with OKEN Scanner

```
        Parametri:
                - matrA: numele primei matrice (înmulţirea matricelor
                        nu este comutativă!) ;
                - matrB: numele celei de-a doua matrice;
                - matrC: numele matricei în care se depune rezultatul înmulţirii;
                - nrLiniiA: numărul de linii ale primei matrice;
                - nrColoaneA: numărul de coloane ale primei matrice;
                - nrColoaneB: numărul de coloane ale celei de-a doua matrice;
        Returnează:
                - void;
*/
void inmultMatr ( int *matrA , int *matrB , int *matrC ,
                  int nrLiniiA , int nrColoaneA , int nrColoaneB )
{
 int i , j , k ;
 int sum ;
 for (i=0; i<nrLiniiA; i++)      {
        for (j=0; j<nrColoaneB; j++)     {
                sum = 0 ;
                for (k=0; k<nrColoaneA; k++)
                        sum += ( * ( matrA + nrColoaneA * i + k ) ) *
                                ( * ( matrB + nrColoaneB * k + j ) ) ;
                * ( matrC + nrColoaneB * i + j ) = sum ;
                }
        }
}
/*-------------------------------------------------------------------------------*/
int main ( void )
{
 int *matrA , *matrB , *matrC , nrLiniiA , nrColoaneA , nrColoaneB , i , j ;
 clrscr() ;
 printf("Dimensiunile matricii A :") ;
 scanf ( "%d %d", &nrLiniiA, &nrColoaneA) ;
 printf("\nNumarul de coloane ale matricii B :") ;
 scanf ( "%d", &nrColoaneB) ;
 matrA = (int *) malloc ( nrLiniiA * nrColoaneA * sizeof(int) ) ;
 matrB = (int *) malloc ( nrColoaneA *nrColoaneB * sizeof(int) ) ;
 matrC = (int *) malloc ( nrLiniiA * nrColoaneB * sizeof(int) ) ;
 if ( matrA==NULL || matrB==NULL || matrC==NULL )        {
        printf("Eroare alocare memorie !") ;
        exit(1) ;
```


Scanned with OKEN Scanner

```
        }
  for (i=0; i<nrLiniiA; i++)
        for (j=0; j<nrColoaneA; j++)      {
                printf("matrA[%d][%d] = ",i,j) ;
                scanf ( "%d", matrA + nrColoaneA * i + j ) ;
                }
  for (i=0; i<nrColoaneA; i++)
        for (j=0; j<nrColoaneB; j++)      {
                printf("matrB[%d][%d] = ",i,j) ;
                scanf ( "%d", matrB + nrColoaneB * i + j ) ;
                }
  inmultMatr ( matrA, matrB, matrC, nrLiniiA, nrColoaneA, nrColoaneB) ;
  printf("\nRezultatul inmultirii : \n") ;
  for (i=0; i<nrLiniiA; i++)
        for (j=0; j<nrColoaneB; j++)
          printf("matrC[%d][%d] = %d\n",i,j, * ( matrC + nrColoaneB * i + j ) ) ;
  printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
  getch() ;
  return 0 ;
}
```

Problema 18.1

Determinați modul de memorare al calculatorului pe care lucrați (*low-byte / low-address* - **little endians**, sau *high-byte / high-address* - **big endians**) folosind o uniune de tipul ([4]):

```
union test {
        int x ;
        char y[sizeof(int)] ;       // sau long
        } ;
```

```
#include <stdio.h>
#include <conio.h>
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 union test {
        int x ;
        char y[sizeof(int)] ;
```


Scanned with OKEN Scanner

```
        } a ;
  clrscr() ;
  a.x = 1 ;
  if ( a.y[0] == 1 )
        printf("Modul de adresare este low-byte/low-address ( little endians ).") ;
  else if ( a.y[0] == 0 )
        printf("Modul de adresare este high-byte/low-address ( big endians ).") ;
  printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
  getch() ;
  return 0 ;
}
```

Problema 18.2

Se consideră structura de tipul

```
struct cmpl {
        double real ;     // partea reală
        double imag ;     // partea imaginară
        } ;
```

pentru lucrul cu numere complexe (*Turbo C++* oferă toate operațiile cu numere complexe prin intermediul structurii *complex*, definită în **MATH.H** - în programele C++).

Scrieți pentru operațiile de adunare şi respectiv de înmulțire a numerelor complexe cîte două funcții: una în varianta pre-ANSI (se acceptă transmiterea structurilor ca parametru ori valoare de retur numai prin adresă) şi alta în varianta ANSI (se acceptă transmiterea structurilor ca un tot unitar).

Atenție: transmiterea structurilor ca parametru încarcă (de regulă excesiv) stiva!

```
#include <stdio.h>
#include <conio.h>
#include <float.h>
typedef struct    {
        double real ;
        double imag ;
        } CMP ;
/*-------------------------------------------------------------------------*/
/*
```


Scanned with OKEN Scanner

Funcția **sumaAnsi** calculează și returnează suma a doua numere complexe emulate prin intermediul unor structuri de tip CMP.

Parametri:

- *x*: numerele care se adună;
- *y*

Returnează:

- *CMP*: suma celor două numere transmise ca parametri;

```
*/
CMP sumaAnsi ( CMP x , CMP y )
{
        CMP z ;
        z.real = x.real + y.real ;
        z.imag = x.imag + y.imag ;
        return z ;
}
/*-------------------------------------------------------------------------------------------*/
/*
```

Funcția **sumaPreAnsi** calculează și returnează suma a două numere complexe emulate prin intermediul unor structuri de tip CMP, folosind transmiterea structurilor prin adresă.

Parametri:

- *x*: adresele numerelor care se adună;
- *y*
- *z*: adresa rezultatului adunării;

Returnează:

- *void*;

```
*/
void sumaPreAnsi ( CMP *x , CMP *y , CMP *z )
{
        z->real = x->real + y->real ;
        z->imag = x->imag + y->imag ;
}
/*-------------------------------------------------------------------------------------------*/
/*
```

Funcția **produsAnsi** calculează și returnează produsul a doua numere complexe emulate prin intermediul unor structuri de tip CMP.

Parametri:

- *x*: numerele care se adună;
- *y*

Returnează:

- *CMP*: produsul celor două numere transmise ca parametri;

Scanned with OKEN Scanner

```
*/
CMP produsAnsi ( CMP x , CMP y )
{
        CMP z ;
        z.real = x.real*y.real - x.imag*y.imag ;
        z.imag = x.imag*y.real + x.real*y.imag ;
        return z ;
}
/*-------------------------------------------------------------------------------------*/
/*
```

Funcția **produsPreAnsi** calculează și returnează produsul a două numere complexe emulate prin intermediul unor structuri de tip CMP, folosind transmiterea structurilor prin adresă.

Parametri:

- *x*: adresele numerelor care se adună;
- *y*
- *z*: adresa rezultatului înmulțirii;

Returnează:

- *void*;

```
*/
void produsPreAnsi ( CMP *x , CMP *y , CMP *z )
{
        z->real = x->real*y->real - x->imag*y->imag ;
        z->imag = x->imag*y->real + x->real*y->imag ;
}
/*-------------------------------------------------------------------------------------*/
int main(void)
{
 CMP a, b, s1, s2, p1, p2;
 clrscr() ;
 printf("\nPartea reala a primului numar complex : ");
 scanf("%lf",&a.real) ;
 printf("\nPartea imaginara a primului numar complex : ");
 scanf("%lf",&a.imag) ;
 printf("\nPartea reala a celui de-al doilea numar complex : ");
 scanf("%lf",&b.real) ;
 printf("\nPartea imaginara a celui de-al doilea numar complex : ");
 scanf("%lf",&b.imag) ;
 s1 = sumaAnsi ( a, b );
 sumaPreAnsi ( &a, &b, &s2 );
 p1 = produsAnsi ( a, b );
```


Scanned with OKEN Scanner

```
  produsPreAnsi ( &a, &b, &p2 );
  printf("\n\nSuma     pre-ANSI : %.3lf %c j*%.3lf", s1.real,
                  s1.imag >= 0 ? '+' : '-',
                  s1.imag >= 0 ? s1.imag : -s1.imag ) ;
  printf("\n\nSuma     ANSI     : %.3lf %c j*%.3lf", s2.real,
                  s2.imag >= 0 ? '+' : '-',
                   s2.imag >= 0 ? s2.imag : -s2.imag ) ;
  printf("\n\nProdusul pre-ANSI : %.3lf %c j*%.3lf", p1.real,
                  p1.imag >= 0 ? '+' : '-',
                   p1.imag >= 0 ? p1.imag : -p1.imag ) ;
  printf("\n\nProdusul ANSI     : %.3lf %c j*%.3lf", p2.real,
                  p2.imag >= 0 ? '+' : '-',
                   p2.imag >= 0 ? p2.imag : -p2.imag ) ;
  printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
  getch() ;
  return 0 ;
}
```

Problema 18.3

Scrieţi o funcţie cu o listă variabilă de argumente care să realizeze interclasarea optimală a unor tablouri sortate de numere întregi.

Pentru optim se vor interclasa la fiecare pas cele două tablouri cu cele mai puţine elemente, pînă cînd rămîn numai două, a căror interclasare va da rezultatul final ([5], pag. 96).

Adoptaţi orice convenţie simplificatoare vă avantajează (de exemplu valoarea 0 să marcheze sfîrşitul fiecărui tablou de întregi, pentru a nu mai trebui ca numărul de elemente al fiecărui tablou să fie transmis ca parametru).

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <stdarg.h>
/*-------------------------------------------------------------------------------------------*/
/*
```

Funcţia **lungimeTablou** calculează şi returnează numărul elementelor dintr-un tablou de întregi - convenţia este că 0 reprezintă ultimul element (nu se include în sortări).

Parametri:

- *a*: numele tabloului;

Scanned with OKEN Scanner

```
        Returnează:
                - int: numărul elementelor din tablou;
*/
int lungimeTablou ( int *a )
{
 int i = 0 ;
 while ( a[i]!=0 )          i++ ;
 return (i+1) ;
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcția lungComp se folosește pentru compararea lungimilor a două
tablouri de întregi conform convenției amintite - se va folosi în qsort().
        Parametri:
                - a: numele tablourilor implicate;
                - b
        Returnează:
                - int: rezultatul comparării;
*/
int lungComp ( int **a , int **b )
{
 return ( lungimeTablou(*a) - lungimeTablou(*b) ) ;
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcția interclas realizează interclasarea a două tablouri de întregi ordonate
crescător.
        Parametri:
                - a,b: numele tablourilor de interclasat;
                - c: numele tabloului în care se depune rezultatul;
        Returnează:
                - void;
*/
void interclas ( int *a , int *b , int *c )
{
        int x , y , z ;
        x = y = z = 0 ;
        while ( a[x] != 0 && b[y] != 0 )
                if ( a[x] < b[y] )
                        c[z++] = a[x++] ;
                else
```


Scanned with OKEN Scanner

```
                    c[z++] = b[y++] ;
        if ( a[x] == 0 )
                while ( b[y] != 0 )
                        c[z++] = b[y++] ;
        else
                while ( a[x] != 0 )
                        c[z++] = a[x++] ;
        c[z] = 0 ;
}
/*-----------------------------------------------------------------------------------------*/
/*
```

Funcția **eroareAlocare** semnalează imposibilitatea alocării memoriei cerute și abandonează programul:

Parametri:

- *void*;

Returnează:

- *void*;

```
*/
void eroareAlocare ( void )
{
        printf("\n\aEroare alocare memorie !") ;
        exit(1) ;
}
/*-----------------------------------------------------------------------------------------*/
/*
```

Funcția **interclasOptimala** realizează interclasarea optimală a mai multor tabele de întregi.

Parametri:

- *fictiv*: șir de caractere folosit pentru testare și 'pornirea' listei variabile de argumente;

Returnează:

- *int* *: tabelul final rezultat;

ATENȚIE: alterează tablourile ce i se transmit ca parametri !!!

```
*/
int *interclasOptimala ( char *fictiv , ... )
{
 int *rezultat ;
 int **temp ;
 int *arg ;
 int numarTabele ;
 int **manevra ;
```


Scanned with OKEN Scanner

```
int i = 0 ;
va_list ap ;
printf("\nOPTIM : %s\n",fictiv ) ;
                        // Se alocă spaţiu pentru 10 parametri.
if ( ( temp = (int **) malloc ( 10 * sizeof(int *) ) ) == NULL )
        eroareAlocare() ;
va_start ( ap , fictiv ) ;
while ( ( arg = va_arg ( ap , int *) ) != NULL ) {
        if ( i!=0 && i%10==0 ) {                    // încă mai sînt parametri
                temp = (int **) realloc ( temp, (i+10) * sizeof(int *) ) ;
                if ( temp == NULL )
                        eroareAlocare() ;
                }
        temp[i++] = arg ;
        }
va_end ( arg ) ;
numarTabele = i ;
if ( numarTabele == 0 )         {
        free(temp) ;
        return NULL ;
        }
if ( numarTabele == 1 )         {
        rezultat = temp[0] ;
        free(temp) ;
        return rezultat ;
        }
manevra = (int **) malloc ( i * sizeof(int *) ) ;
if ( manevra == NULL )
        eroareAlocare() ;
                // Se ordonează lista numelor tablourilor după numărul de elemente.
qsort ( temp, numarTabele, sizeof(int *),
          (int (*)(const void *,const void *))lungComp ) ;
                // La fiecare pas se vor sorta tablourile cu cele mai puţine elemente.
for (i=0; i<numarTabele-2; i++)         {
 manevra[i] = (int *)malloc( ( lungimeTablou(temp[i]) +
                                lungimeTablou(temp[i+1] ) -1) * sizeof(int) ) ;
 if ( manevra[i] == NULL )
       eroareAlocare() ;
 interclas ( temp[i] , temp[i+1] , manevra[i] ) ;
 temp[i+1] = manevra[i] ;
                // Se reordonează tablourile rămase.
```


Scanned with OKEN Scanner

```
    qsort ( &(temp[i+1]), numarTabele - i - 1, sizeof(int *),
                (int (*)(const void *,const void *))lungComp ) ;
    }
  rezultat = (int *) malloc ( ( lungimeTablou( temp[numarTabele-2] ) +
                               lungimeTablou ( temp[numarTabele-1] ) - 1 )
                               * sizeof(int) ) ;
  if ( rezultat == NULL )
        eroareAlocare() ;
  interclas ( temp[numarTabele-2] , temp[numarTabele-1] , rezultat ) ;
  for (i=0; i<=numarTabele-3; i++)
        free(manevra[i]) ;
  free ( temp ) ;
  free ( manevra ) ;
  return rezultat  ;
}
                // Se testează cu nişte tablouri de întregi predefinite, gata sortate.
int a[] = { 1 , 3  , 5 , 8 , 12 , 0 } ;
int b[] = { 2 , 3 , 4 , 7 , 0  } ;
int c[] = { 3 , 22 , 33 , 44 , 45 , 46 , 0 } ;
int d[] = { 1 , 2 , 4 , 6 , 14 , 22 , 33 , 0 } ;
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 int *final , i ;
 clrscr() ;
 printf("\nIncepe rularea :\n") ;
 final = interclasOptimala ( "A pornit !\n" , a , b , c , d , NULL ) ;
 printf("\nREZULTAT FINAL :\n") ;
 if ( final == NULL )
        printf("\nLipsa combatanti ...\n") ;
 else
        for (i=0; final[i]!=0; i++)
                printf ( "%3d -%c", final[i], i%10 == 9 ? '\n' : ' ' ) ;
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 19.1

Scanned with OKEN Scanner

Să se scrie un program C care creează o listă simplu înlănțuită, avînd ca parte de informație un număr întreg. Elementele vor fi inserate în listă cîte o singură dată, în ordinea crescătoare a părților de informație.

Programul va include și o funcție de ștergere a unui element (indicat prin partea de informație) din listă.

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#include <process.h>
typedef struct lista        {
        int inf ;
        struct lista *urm ;
        }       LIS ;
void afisareLista ( LIS * ) ;
int stergereLista ( int , LIS **) ;
/*-------------------------------------------------------------------------------------------*/
/*
        Funcția inserareLista inserează un nou element în listă, conform condițiilor
enunțate.
        Parametri:
                - u: partea de informație a noului element;
                - v: adresa capului listei;
        Returnează:
                - void;
*/
void inserareLista ( int u , LIS **v )
{
 LIS *a , *b , *c ;
 if ( ( a = (LIS *) malloc ( sizeof(LIS) ) ) == NULL )      {
        printf("\nNu poate fi alocata memorie !\a") ;
        exit(1) ;
        }
 a->inf = u ;
 a->urm = NULL ;
 if ( *v == NULL )      {                       // lista era vidă
        *v = a ;
        return;
        }
 for ( b=*v; b!=NULL && b->inf<u; b=b->urm )
        c = b ; // se memorează adresa precedentului element
```


Scanned with OKEN Scanner

```
if ( b == NULL )        // se face inserarea în coada listei
        c->urm = a ;
else if ( b->inf == u )  {      // exista deja un asemenea element
        free(a) ;
        printf("\n\aExista deja !\n");
        }
else if ( b == *v )  {      // se face inserarea înaintea actualului cap
        a->urm = *v     ;
        *v=a;
        }
else  {
        a->urm = c->urm ;
        c->urm = a ;
        }
}
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 int x ;
 LIS *cap = NULL ;
 clrscr() ;
 printf("Primul element al listei : ") ;
 scanf("%d",&x) ;
 while ( x != 0 ){
        inserareLista ( x, &cap ) ;
        printf("Urmatorul element ( 0 pentru incheiere ) : ") ;
        scanf("%d",&x) ;
        }
 printf("\nContinutul listei :\n") ;
 afisareLista ( cap ) ;
 while(1)        {
        printf("\nElementul de sters ( 0 pentru incheiere ) :") ;
        scanf("%d",&x) ;
        if ( x==0 )
                break ;
        if ( stergereLista(x,&cap) )
                printf("\n%d nu se afla in lista !",x) ;
        else    {
                printf("\nLista actualizata :\n") ;
                afisareLista(cap) ;
                }
```


Scanned with OKEN Scanner

```
        if ( cap==NULL )
                break ;
        }
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia afisareLista afişează conţinutul listei.
        Parametri:
                - a: capul listei;
        Returnează:
                - void;
*/
void afisareLista ( LIS *a )
{
  if ( a == NULL )      {
        printf("\nLista este vida !\a") ;
        return ;
        }
  while ( a != NULL )   {
        printf("%d %s ", a->inf, a->urm != NULL ? "->" : "" ) ;
        a = a->urm ;
        }
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcţia stergereLista şterge un nou element din listă, conform condiţiilor
enunţate.
        Parametri:
                - u: partea de informaţie a elementului de şters;
                - v: adresa capului listei;
        Returnează:
                - int: 0 pentru succes, 1 dacă nu există un asemenea element;
*/
int stergereLista ( int u , LIS **v )
{
 LIS *a , *b ;
 for ( a=*v; a!=NULL && a->inf<u ; a=a->urm)
        b = a ;
```


Scanned with OKEN Scanner

```
if ( a==NULL || a->inf!=u)
        return 1 ;
if ( a == *v )              // se şterge elementul din cap
        *v = a->urm ;
else
        b->urm = a->urm ;
free(a) ;
return 0 ;
}
```

Problema 19.2

Creaţi alte două funcţii pentru inserare / ştergere (corespunzătoare programului precedent) de forma:

```
LIS *insert ( int u, LIS *v ) ;
LIS *delete ( int u, LIS *v ) ;
```

De această dată *v* va fi capul listei iar funcţiile vor returna noul cap (eventual modificat); apelurile vor fi de forma:

```
cap = insert ( x, cap ) ;
```

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#include <process.h>
typedef struct lista        {
        int inf ;
        struct lista *urm ;
        }       LIS ;
void afisareLista ( LIS *x ) ;
LIS *stergereLista ( int u , LIS *v , int *exista ) ;
/*-------------------------------------------------------------------------------*/
/*
        Funcţia inserareLista inserează un nou element în listă, conform condiţiilor
enunţate.
        Parametri:
                - u: partea de informaţie a noului element;
                - v: capul listei;
        Returnează:
```


Scanned with OKEN Scanner

- *LIS* *: capul listei după inserare:

```
*/
LIS *inserareLista ( int u , LIS *v )
{
 LIS *a , *b , *c ;
 if ( ( a = (LIS *) malloc ( sizeof(LIS) ) ) == NULL )     {
        printf("\nNu poate fi alocata memorie !\a") ;
        exit(1) ;
        }
 a->inf = u ;
 a->urm = NULL ;
 if ( v == NULL )                          // lista era vidă
        return a ;
 for ( b=v; b!=NULL && b->inf<u; b=b->urm )
        c = b ;
 if ( b == NULL )
        c->urm = a ;
 else if ( b->inf == u )   {
        printf("\n\aExista deja !\n");
        free(a) ;
        }
 else if ( b == v )          {
        a->urm = v ;
        return a ;
        }
  else  {
        a->urm = c->urm ;
        c->urm = a ;
        }
 return v ;
}
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 int x , exista ;
 LIS *cap = NULL ;
 clrscr() ;
 printf("Primul element al listei : ") ;
 scanf("%d",&x) ;
 while ( x != 0 ){
        cap = inserareLista ( x,-cap ) ,
```


Scanned with OKEN Scanner

```
        printf("Urmatorul element ( 0 pentru incheiere ) : ") ;
        scanf("%d",&x) ;
        }
printf("\nContinutul listei :\n") ;
afisareLista(cap) ;
while(1)        {
        printf("\nElementul de sters :") ;
        scanf("%d",&x) ;
        if ( x==0 )
                break ;
        exista = 1 ;
        cap = stergereLista ( x , cap , &exista ) ;
        if ( !exista )
                printf("\n%d nu se afla in lista !\a",x) ;
        else            {
                printf("\nLista actualizata :\n") ;
                afisareLista(cap) ;
                }
        if ( cap == NULL )
                break ;
        }
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
/*-------------------------------------------------------------------------------------*/
/*
        Funcția afisareLista afișcază conținutul listei.
        Parametri:
                - a: capul listei;
        Returnează:
                - void;
*/
void afisareLista ( LIS *a )
{
 if ( a == NULL )       {
        printf("\nLista este vida !\a") ;
        return ;
        }
 while ( a != NULL )    {
        printf("%d %s ", a->inf, a->urm != NULL ? "->" : "" ) ;
```


Scanned with OKEN Scanner

```
        a = a->urm ;
        }
}
/*-------------------------------------------------------------------------------*/
/*
```

Funcția **stergereLista** șterge un nou element din listă, conform condițiilor enunțate.

Parametri:

- *u*: partea de informație a elementului de șters;
- *v*: capul listei;
- *exista*: adresa unei variabile ce semnalează inexistența elementului vizat în listă;

Returnează:

- *LIS.**: capul listei dupa ștergere;

```
*/
LIS *stergereLista ( int u , LIS *v , int *exista )
{
 LIS *a , *b , *rezultat = v ;
 for ( a=v; a!=NULL && a->inf<u ; a=a->urm )
        b = a ;
 if ( a==NULL || a->inf!=u )                {
        *exista = 0 ;
        return v ;
        }
 if ( a==v )                // se șterge elementul din capul listei
        rezultat = a->urm ;
 else
        b->urm = a->urm ;
 free(a) ;
 return rezultat ;
}
```

Problema 19.3

Scrieți un program care creează o listă dublu înlănțuită, avînd ca parte de informație șiruri de caractere.

Elementele listei vor fi de forma :

```
struct listaDubla          {
        struct listaDubla *ant ;
```


Scanned with OKEN Scanner

```
        char *sir ;
        struct listaDubla *post ;
        } ;
```

Lista creată va fi tot sortată, fiecare element apărînd o singură dată.

Doi parametri opţionali în linia de comandă vor impune ca să nu se diferenţieze literele mari de cele mici, respectiv ca ordonarea să se facă crescător sau descrescător (implicit : crescător, făcînd diferenţierea literelor mari de cele mici).

Programul va include şi o funcţie de ştergere a unui element din listă.

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#include <string.h>
#include <process.h>
typedef struct listaDubla {
        struct listaDubla *ant ;
        char *sir ;
        struct listaDubla *post ;
        }       LISD ;
void afisareLista ( LISD *x ) ;
void inserareLista ( char *u , LISD **v , int nuDe , int cresc ) ;
int stergereLista ( char *u , LISD **v , int nuDe , int cresc ) ;
/*-------------------------------------------------------------------------------------*/
int main ( int argc , char *argv[] )
/*
                Se apelează de forma : 19_3 [/I] [/D]
*/
{
 char tab[80] ;
 int nuDeosob = 0 ,  // nu se deosebesc literele mici de cele mari dacă are valoarea 0
     cresc = 1 ;     // ordonare crescătoare (descrescatoare dacă are valoarea 0)
 LISD *cap = NULL ;
 clrscr() ;
 if ( argc>1 )   {
        if ( !stricmp(argv[1],"/I") )
                nuDeosob = 1 ;
        else if ( !stricmp(argv[1],"/D"))
                cresc = 0 ;
        }
 if ( argc>2 )   {
```


Scanned with OKEN Scanner

```
        if ( !stricmp(argv[2],"/I") )
                nuDeoseb = 1 ;
        else if ( !stricmp(argv[2],"/D") )
                cresc = 0 ;
        }
printf("\nSe incheie inserarea si stergerea cu o linie vida.\n");
while ( printf("Noul element : ") , gets(tab) , strlen(tab) )
        inserareLista ( tab , &cap , nuDeoseb , cresc ) ;
printf("\nLista creata :\n") ;
afisareLista(cap) ;
while(1)        {
        printf("\nElementul de sters : ") ;
        gets(tab) ;
        if ( !strlen(tab) )             break ;
        if (stergereLista(tab,&cap,nuDeoseb,cresc))
                printf("\n\a\"%s\" nu se afla in lista",tab) ;
        else    {
                printf("\nLista actualizata :\n") ;
                afisareLista(cap) ;
                }
        if ( cap == NULL )              break ;
        }
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
/*-------------------------------------------------------------------------------*/
/*
        Funcţia afisareLista afişează conţinutul listei.
        Parametri:
                - a: capul listei;
        Returnează:
                - void;
*/
void afisareLista ( LISD *a )
{
 if ( a == NULL )               {
        printf("\nLista este vida !") ;
        return ;
        }
 while ( a != NULL )    {
```


Scanned with OKEN Scanner

```
        printf("\t\t%s\n",a->sir) ;
        a = a->post ;
        }
}
/*-------------------------------------------------------------------------------------------*/
/*
```

Funcția **inserareLista** inserează un nou element în listă, conform condițiilor enunțate.

Parametri:

- *u*: partea de informație a noului element;
- *v*: adresa capului listei;
- *nuDe*: 1 dacă nu se deosebesc literele mari de cele mici la comparare, altfel 0;
- *cresc*: 1 dacă se face ordonare (lexicografic) crescătoare, altfel 0;

Returnează:

- *void*;

```
*/
void inserareLista ( char *u , LISD **v , int nuDe , int cresc )
{
 LISD *a, *b , *c ;
 if ( ( a = (LISD *) malloc ( sizeof(LISD) ) ) == NULL ) {
        printf("\nNu poate fi alocata memorie !\a") ;
        exit(1) ;
        }
                        // Trebuie alocată apoi memorie pentru şirul de caractere !
 if ( ( a->sir = (char *) malloc ( strlen(u)+1) ) == NULL )          {
        printf("\nNu poate fi alocata memorie !\a") ;
        exit(1) ;
        }
 a->ant = a->post = NULL ;
 strcpy ( a->sir, u ) ;
 if ( *v == NULL )  {                    // lista era vidă
        *v = a ;
        return ;
        }
 for (b=*v; b!=NULL &&
            ( nuDe ? ( cresc ? stricmp(b->sir,u)<0 : stricmp(b->sir,u)>0 )
                     : ( cresc ? strcmp(b->sir,u)<0  : strcmp(b->sir,u)>0 ) )
                                : b=b->post)
                                                        c = b ;
 if ( nuDe ? stricmp(b->sir,u)==0 : strcmp(b->sir,u)==0 )          {
```


Scanned with OKEN Scanner

```
        free(a->sir) ;
        free(a) ;
        }
 else if ( b == *v )        {        // inserare înaintea vechiului cap
        a->post = b ;
        b->ant = a ;
        *v = a ;
        }
 else if (b==NULL)      {
        c->post = a ;
        a->ant = c ;
        }
 else   {
        a->ant = c ;
        b->ant = a ;
        c->post = a ;
        a->post = b ;
        }
}
/*-------------------------------------------------------------------------------------------*/
/*
```

Funcția **stergereLista** șterge un nou element din listă, conform condițiilor enunțate.

Parametri:

- *u*: partea de informație a elementului de șters;
- *v*: capul listei;
- *nuDe*: 1 dacă nu se deosebesc literele mari de cele mici la comparare, altfel 0;
- *cresc*: 1 dacă se face ordonare (lexicografic) crescătoare, altfel 0;

Returnează:

- *int*: 1 dacă elementul indicat nu există, altfel 0;

```
*/
int stergereLista ( char *u , LISD **v , int nuDe , int cresc )
{
 LISD *a , *b ;
 for (a=*v; a!=NULL &&
                ( nuDe ? ( cresc ? stricmp(a->sir,u)<0 : stricmp(a->sir,u)>0 )
                        : ( cresc ? strcmp(a->sir,u)<0 : strcmp(a->sir,u)>0 ) ) ;
                                        a=a->post)
                                                b = a ;
 if ( a==NULL || ( nuDe ? stricmp(a->sir,u) : strcmp(a->sir,u) ) )
```


Scanned with OKEN Scanner

```
        return 1 ;
if ( a == *v )   {
        b = a->post ;
        if (b!=NULL)
                b->ant = NULL ;
        *v = b ;
        }
else    {
        b->post = a->post ;
        if (a->post!=NULL)
                (a->post)->ant = b ;
        }
 free(a->sir) ;
 free(a) ;
 return 0 ;
}
```

Problema 19.4

Să se scrie un program care creează două liste simplu înlănțuite (sortate: 19.1) avînd ca parte de informație numere întregi, după care le interclasează și afișează rezultatul.

Lista rezultat se va compune (fizic !) din elementele celor două lsite; un element va apărea tot o singură dată.

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#include <process.h>
typedef struct lista        {
        int inf ;
        struct lista *urm ;
        } LIS ;
// Se refolosesc funcțiile prezentate la problema 19.1 (nu se mai include codul).
/*-------------------------------------------------------------------------------*/
/*
```

Funcția **interclasListe** realizează interclasarea a două liste sortate conform criteriilor descrise.

Parametri:

- *a,b*: capurile celor doua liste de interclasat;

Scanned with OKEN Scanner

```
        Returnează:
                - LIS *: capul listei rezultate;
*/
LIS *interclasListe ( LIS *a , LIS *b )
{
 LIS *t , *rez , *c ;
                // Se tratează cazurile triviale - vreuna din liste este vidă.
 if ( a == NULL )
        return b ;
 else if ( b == NULL )
        return a ;
         // Se stabileşte capul listei rezultate din interclasare (valoarea de returnat).
 if ( a->inf < b->inf )    {
        rez = a ;
        a = a->urm ;
        }
 else if ( a->inf==b->inf )        {
        rez = a ;
        a = a->urm ;
        t = b ;
        b = b->urm ;
        free(t) ;
        }
 else    {
        rez = b ;
        b = b->urm ;
        }
 c = rez ;
 while ( a!=NULL && b!=NULL )       {
        if ( a->inf < b->inf )     {
                c->urm = a ;
                c = c->urm ;
                a = a->urm ;
                }
        else if ( a->inf > b->inf ) {
                c->urm = b ;
                c = c->urm ;
                b = b->urm ;
                }
        else    {
                c->urm = b ;
```


Scanned with OKEN Scanner

```
                c = c->urm ;
                b = b->urm ;
                t = a ;
                a = a->urm ;
                free(t) ;
                }
        }
if ( a==NULL )          // s-a terminat prima listă
        while ( b!=NULL )               {
                c->urm = b ;
                c = c->urm ;
                b = b->urm ;
                }
else
        while ( a!=NULL )               {
                c->urm = a ;
                c = c->urm ;
                a = a->urm ;
                }
return rez ;
}
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
int x ;
LIS *cap1 = NULL , *cap2 = NULL , *cap3 ;
clrscr() ;
printf("Primul element al listei 1 : ") ;
scanf("%d",&x) ;
while (x!=0)    {
        inserareLista(x,&cap1) ;
        printf("Urmatorul element - 1 ( 0 pentru incheiere ) : ") ;
        scanf("%d",&x) ;
        }
printf("\n\nPrimul element al listei 2 : ") ;
scanf("%d",&x) ;
while (x!=0)    {
        inserareLista(x,&cap2) ;
        printf("Urmatorul element - 2 ( 0 pentru incheiere ) : ") ;
        scanf("%d",&x) ;
        }
```


Scanned with OKEN Scanner

```
 printf("\nContinutul listei 1 :\n") ;
 afisareLista(cap1) ;
 printf("\nContinutul listei 2 :\n") ;
 afisareLista(cap2) ;
 cap3 = interclasListe ( cap1 , cap2 ) ;
 printf("\nRezultatul interclasarii :\n") ;
 afisareLista(cap3) ;
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 19.7

Reluați problema lui *Josephus* (**6.3**), folosind de această dată o listă circulară.

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#include <stdlib.h>
typedef struct JOSEPHUS  {
        int numarOrdine ;
        struct JOSEPHUS *urm ;
        }  J ;
/*-------------------------------------------------------------------------------------*/
int main ( void )
{
 int n , k , numarRamase , i , nr ;
 J *start = NULL , *a , *b = NULL ;
 clrscr() ;
 printf("\nIntroduceti N si K :") ;
 scanf ( "%d%d", &n, &k) ;
 if ( n < 2 || k < 2 )       {
        printf("\n\aValori incorecte ( minimum 2 ) !") ;
        exit(1) ;
        }
 numarRamase = n ;
                        // Se creează lista circulară corespunzătoare.
 for ( nr = 0 ; nr<n ; nr++ )  {
```


Scanned with OKEN Scanner

```
        if ( ( a = (J *) malloc ( sizeof(J) ) ) == NULL )    {
                printf("\n\aEroare alocare memorie !") ;
                exit(1) ;
                }
        if ( !nr ) start = a ;
        a->numarOrdine = nr + 1 ;
        if ( b != NULL )
                b->urm = a ;
        b = a ;
        }
  a->urm = start ;  // se face legătura circulară ( de la ultimul element la cap )
  a = start ;
  while ( numarRamase > 1 )      {
        for ( i=1; i<k-1; i++ )
                a = a->urm ;
        b = a->urm ;                // persoana de eliminat
        a->urm = b->urm ;
        a = a->urm ;
/*
        printf("\nSe elimina persoana %d ...",b->numarOrdine) ;
        getch() ;
*/
        free(b) ;
        numarRamase-- ;
        }
  printf("\nRamine : %d ...",a->numarOrdine) ;
  printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
  free(a) ;
  getch() ;
  return 0 ;
}
```

Problema 21.1

Să se scrie un program care citeşte de la tastatură data calendaristică dorită (sub forma **ll/zz/aa[aa]**, cu cifre pentru lună-zi-an, sau **zz-mmm-aa[aa]**, cu cifre pentru zi-an şi trei caractere pentru numele lunii - varianta română !), fixează data sistem, o afişează apelînd comanda DOS *DATE*, după care se termină refăcînd data iniţială.

Scanned with OKEN Scanner

Programul va încerca pe rînd interpretarea unui şir de caractere din intrare în ambele variante, fără să se indice în vreun fel formatul folosit.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <dos.h>
#include <string.h>
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 struct date xx , yy ;
 char *numeLuni[] = { "ian" , "feb" , "mar" , "apr" , "mai" , "iun" , "iul" , "aug" ,
                      "sep" , "oct" , "nov" , "dec" } ;
 int lungimiLuni[] = { 31 , 28 , 31 , 30 , 31 , 30 , 31 , 31 , 30 , 31 , 30 , 31 } ;
 int numarZi , nrLuna , numarAn , i , bisect ;
 char tab[30] , l[4] ;
 clrscr() ;
 printf("Introduceti data dorita  ( ll/zz/aa[aa] sau zz-lll-aa[aa] ) : ") ;
 gets(tab) ;
                  // sscanf() returnează numărul elementelor corect convertite.
 if ( sscanf ( tab, "%2d/%2d/%4d", &numarZi, &nrLuna, &numarAn ) != 3 )
   if ( sscanf ( tab, "%2d-%3s-%4d", &numarZi, l, &numarAn) == 3 )    {
        for (i=0; i<12; i++)
          if ( stricmp(l,numeLuni[i]) == 0 )                    break ;
          if ( i == 12 )  {
                printf("\n\n\aNumele lunii ne este necunoscut ...") ;
                exit(1) ;
                }
          nrLuna = i+1 ;
          }
   else {
     printf("\n\n\aFormatul corect este  dd/ll/[aa]aa sau dd-lll-[aa]aa  !") ;
     exit(1) ;
     }
nrLuna-- ;
                        // DOS lucrează în intervalul 1980 ÷ 2079.
if ( numarAn >= 80 && numarAn <= 99 )
        numarAn += 1900 ;
else if ( numarAn >= 0 && numarAn < 80 )
        numarAn += 2000 ;
```


Scanned with OKEN Scanner

```
if ( numarAn < 1980 || numarAn > 2079 )        {
        printf("\n\n\aDOS cunoaste numai intervalul 1980-2079 !") ;
        exit(1) ;
        }
if ( nrLuna<0 || nrLuna>=12 )  {
        printf("\n\n\aExista nrLuna numarul %d ?! ",nrLuna) ;
        exit(1) ;
        }
bisect = numarAn%4==0 && numarAn%100!=0 || numarAn%400==0 ;
lungimiLuni[1] += bisect ;
                // dacă este un an bisect, se modifică lungimea lunii februarie
if ( numarZi<1 || numarZi>lungimiLuni[nrLuna] )        {
        printf("\n\n\aAre luna %s ziua cu numarul %d in anul %d ?!",
                numeLuni[nrLuna] , numarZi , numarAn ) ;
        exit(1) ;
        }
printf("\nInainte de schimbarea datei prin program :\n") ;
if ( system("DATE") )
        perror("\nLansare I - comanda DATE  ") ;
getdate(&xx) ;                          // memorarea datei inițiale
yy.da_year = numarAn ;                  // pregătirea structurii yy
yy.da_day  = numarZi ;
yy.da_mon  = nrLuna + 1 ;
setdate(&yy) ;
printf("\nDupa schimbarea datei prin program :\n") ;
if ( system("DATE") )
      perror("\nLansare II - comanda DATE  ") ;
printf("\n\nSe reface data initiala . \n") ;
setdate(&xx) ;
if (system("DATE"))
        perror("\nLansare III -  comanda DATE  ") ;
printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
getch() ;
return 0 ;
}
```

Problema 21.4

Scrieți un program care citește de la tastatură o dată calendaristică, după care afișează a cîta zi din este și în ce zi a săptămînii cade (pentru a putea folosi

Scanned with OKEN Scanner

funcțiile prezentate va trebui să ne încadrăm în intervalul 1980 ÷ 2079, altfel trebuind să creăm funcții proprii).

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <dos.h>
#include <string.h>
#include <time.h>
/*-------------------------------------------------------------------------------*/
int main ( void )
{
 struct tm xx ;
 char *numeLuni[] = { "ian" , "feb" , "mar" , "apr" , "mai" , "iun" , "iul" , "aug" ,
                      "sep" , "oct" , "nov" , "dec" } ;
 int lungimiLuni[] = { 31 , 28 , 31 , 30 , 31 , 30 , 31 , 31 , 30 , 31 , 30 , 31 } ;
 char *numeZile[] = { "duminica" , "luni" , "marti" , "miercuri" , "joi" , "vineri" ,
                      "simbata"  } ;
 int numarZi , numarLuna , numarAn , i , bisect ;
 char tab[30] , l[4] ;
 clrscr() ;
 printf("Introduceti data dorita ( ll/zz/aa[aa] sau zz-lll-aa[aa] ) : ") ;
 gets(tab) ;
 if ( sscanf ( tab, "%2d/%2d/%4d", &numarZi, &numarLuna, &numarAn) != 3 )
   if ( sscanf ( tab, "%2d-%3s-%4d", &numarZi, l, &numarAn) == 3 )    {
        for (i=0; i<12; i++)
                if (stricmp(l,numeLuni[i])==0)
                                break ;
        if (i==12)        {
                printf("\n\n\aNumele lunii ne este necunoscut ...") ;
                exit(1) ;
                }
        numarLuna = i+1 ;
        }
   else {
        printf("\n\n\aFormatul corect este  dd/ll/[aa]aa sau dd-lll-[aa]aa  !") ;
        exit(1) ;
        }
 numarLuna-- ;
 if ( numarAn>=80 && numarAn<=99 )
        numarAn += 1900 ;
```


Scanned with OKEN Scanner

```
else if ( numarAn>=0 && numarAn<80 )
        numarAn += 2000 ;
if ( numarAn<1980 || numarAn>2079 ) {
        printf("\n\n\aDOS cunoaste numai intervalul 1980-2079 !") ;
        exit(1) ;
        }
if ( numarLuna<0 || numarLuna>=12 ) {
        printf("\n\n\aExista luna numarul %d ?! ",numarLuna) ;
        exit(1) ;
        }
bisect = numarAn%4==0 && numarAn%100!=0 || numarAn%400==0 ;
lungimiLuni[1] += bisect ;
if ( numarZi<1 || numarZi>lungimiLuni[numarLuna] )  {
        printf("\n\n\aAre luna %s ziua cu numarul %d in anul %d ?!",
                        numeLuni[numarLuna] , numarZi , numarAn ) ;
        exit(1) ;
        }
xx.tm_year  = numarAn - 1900 ;
xx.tm_mon   = numarLuna ;
xx.tm_mday  = numarZi ;
xx.tm_sec   = 0 ;
xx.tm_min   = 0 ;
xx.tm_hour  = 1 ;
xx.tm_isdst = 0 ;
if ( mktime(&xx) == -1 )
        printf("\n\n\a\t Nu se pot determina corect datele cerute ( "
                "poate ati\n\tavansat prea mult in viitor - nu \"merge\" "
                "pina in 2079 ) !") ;
else
        printf("\nData indicata cade intr-o %s si este a %d-a zi a anului .",
                        numeZile[xx.tm_wday] , xx.tm_yday+1 ) ;
printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
getch() ;
return 0 ;
}
```

Problema 21.6

Scrieţi o funcţie avînd un rezultat identic cu **ctime() / asctime()** dar cu datele afişate în limba română.

Scanned with OKEN Scanner

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
/*-------------------------------------------------------------------------------------*/
/*
        Funcția RomCtime este echivalentul "în limba română" pentru funcția de
bibliotecă ctime().
        Parametri:
                - t: adresa unei variabile de tip time_t;
        Returnează:
                - char *: șirul de caractere construit conform cerințelor;
*/
char * RomCtime ( const time_t *t )
{
 static char romtab[30] ;
 char *numeZile[] = { "Dum" , "Lun" , "Mar" , "Mie" , "Joi" , "Vin" , "Sim" , } ;
 char *numeLuni[] = { "Ian" , "Feb" , "Mar" , "Apr" , "Mai" , "Iun" , "Iul" ,
                      "Aug", "Sep" , "Oct" , "Nov" , "Dec" } ;
 struct tm *xx ;
 xx = localtime ( t ) ;
 sprintf ( romtab , "%s %s %02d %02d:%02d:%02d %d\n" ,
                        numeZile[xx->tm_wday] ,
                        numeLuni[xx->tm_mon] ,
                        xx->tm_mday ,
                        xx->tm_hour ,
                        xx->tm_min ,
                        xx->tm_sec ,
                        xx->tm_year + 1900 ) ;
 return romtab ;
}
/*-------------------------------------------------------------------------------------*/
int main ( void )
{
 time_t zz ;
 clrscr() ;
 zz = time ( NULL ) ;
 printf("\n\nIn varianta engleza  : %s", ctime(&zz) ) ;
 printf("\n\nIn varianta romana   : %s", RomCtime(&zz) ) ;
 printf("\n\n\t\tApasati orice tasta pen   terminare...") ;
```


Scanned with OKEN Scanner

```
 getch() ;
 return 0 ;
}
```

Problema 22.1

Să se scrie un program care determină şi afişează eticheta de volum pentru o unitate de disc indicată de la tastatură; pentru comparare, se va lansa din program comanda DOS corespunzătoare (*VOL [d:]*).

```
#include <stdio.h>
#include <conio.h>
#include <io.h>
#include <dir.h>
#include <dos.h>
#include <stdlib.h>
#include <sys\stat.h>
#include <ctype.h>
/*-------------------------------------------------------------------------------------*/
int main(void)
{
  int unitateDorita , unitateCurenta ;
  struct ffblk q ;
  char comanda[50] ;
  clrscr() ;
  unitateCurenta = getdisk() ;
  printf("\nUnitatea dorita : ") ;
  do      {
          unitateDorita = getch() ;
          if ( unitateDorita == 0x1b ) // ESC
                  exit(1) ;
          } while ( unitateDorita<'A' || unitateDorita>'Z' &&
                   unitateDorita<'a' || unitateDorita>'z' ) ;
  printf("%c\n",unitateDorita) ;
  if ( tolower(unitateDorita)-'a' >= setdisk(getdisk()) )       {
          printf("\n\aUnitate eronata !") ;
          exit(1) ;
          }
  setdisk ( tolower ( unitateDorita ) - 'a' ) ;
  if ( !findfirst("\\*.*",&q,FA_LABEL) )
```


Scanned with OKEN Scanner

```
    printf("\nEticheta de volum pentru %c: este %s.\n", toupper(unitateDorita),
                    q.ff_name);
  else
    printf("\nNu are etichetă !\n") ;
  printf("\nComparare cu comanda DOS VOL :\n") ;
  sprintf ( comanda, "VOL %c:", unitateDorita ) ;
  if ( system(comanda) )
          perror("\n\aLansare VOL ") ;
  setdisk(unitateCurenta) ;          // se reface unitatea curentă ca cea inițială
  printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
  getch() ;
  return 0 ;
}
```

Problema 22.2

Testați funcția **searchpath()** pentru a determina și afișa unitatea de disc + directorul în care se află fișierele sistem DOS (MS-DOS sau DR-DOS).

```
#include <stdio.h>
#include <conio.h>
#include <dir.h>
/*-------------------------------------------------------------------------------*/
int main ( void )
{
 char *p , unitateDisc[3] , numeDirector[66] , numeFisier[9] , extensieFisier[5] ;
 char *numeFisiereSistem[] = { "IO.SYS", "MSDOS.SYS", "COMMAND.COM",
                               "IBMBIO.COM" , "IBMDOS.COM" } ;
 int i ;
 clrscr() ;
 for ( i=0; i<5; i++)      {
   p = searchpath(numeFisiereSistem[i]) ;
   if ( p==NULL)
     printf("\n%-15s : nu este gasit .\n", numeFisiereSistem[i]) ;
   else  {
     fnsplit ( p, unitateDisc, numeDirector, numeFisier, extensieFisier ) ;
     printf("\n%-15s : %s%s\n", numeFisiereSistem[i], unitateDisc, numeDirector) ;
     }
   }
printf("\n\n\t\tApasati orice tasta pent  erminare...") ;
```


Scanned with OKEN Scanner

```
 getch() ;
 return 0 ;
}
```

Problema 22.3

Să se scrie un program care parcurge şi afişează toate directoarele (şi subdirectoarele) de pe o unitate de disc indicată de la tastatură, după care se revine pe unitatea iniţială.

> **Atenţie** : toate directoarele (cu excepţia rădăcinii) conţin două intrări speciale - "." pentru directorul curent şi ".." pentru părintele său (sînt şi acestea returnate la căutarea cu **find...()**).

```
#include <stdio.h>
#include <conio.h>
#include <io.h>
#include <dir.h>
#include <dos.h>
#include <stdlib.h>
#include <mem.h>
#include <string.h>
#include <sys\stat.h>
#include <ctype.h>
void parcurgereCompleta(void) ;
/*-------------------------------------------------------------------------------------------*/
int main(void)
{
 int unitateVizata , unitateInitiala ;
 char directorInitial[80] , directorCurent[80] ;
 clrscr() ;
 unitateInitiala = getdisk() ;
 getcwd(directorInitial,80) ;
 printf("\nUnitatea de parcurs : ") ;
 do     {
   unitateVizata = getch() ;
   if (unitateVizata==0x1b)                    exit(1) ;
   }     while ( unitateVizata<'A' || unitateVizata>'Z' &&
                 unitateVizata<'a' || unitateVizata>'z' ) ;
```


Scanned with OKEN Scanner

```
  printf ( "%c\n", toupper(unitateVizata) ) ;
  if ( tolower(unitateVizata) - 'a' >= setdisk(getdisk()) )     {
        printf("\n\aUnitate eronata !") ;
        exit(1) ;
        }
 //  Se trece pe unitatea de disc vizată, în rădăcină ( după memorarea directorului
 // curent corespunzător ).
  setdisk(tolower(unitateVizata)-'a') ;
  getcwd(directorCurent,80) ;
  chdir("\\") ;
  parcurgereCompleta() ;
  chdir(directorCurent) ; // refacerea directorului curent inițial pe unitatea parcursă
  setdisk(unitateInitiala) ;          // refacerea unității de disc inițiale
  chdir(directorInitial) ;  // refacerea directorului curent pentru unitatea inițiala ( în
                                      // caz că aceasta a fost şi cea parcursă )
  printf("\nApasati orice tasta ...") ;
  getch() ;
  return 0 ;
}
/*-------------------------------------------------------------------------------------------*/
/*
        Funcția parcurgereCompleta parcurge o unitate de disc astfel încît să se
treacă prin toate directoarele.
        Parametri:
                - void;
        Returnează:
                - void;
*/
void parcurgereCompleta ( void )
{
  struct ffblk w ;
  char buffer[81] ;
                                      // se afişează directorul parcurs
  printf("%s\n",getcwd(buffer,80)+2) ;  // ca să nu mai apară d:
                                      // +3 ar face invizibilă rădăcina
//  Se exclud simbolurile "." (s-ar schimba directorul în cel curent -> buclă infinită)
// şi ".." (se trece explicit în directorul părinte la terminarea parcurgerii în adîncime).

//  Se "prind" şi directoarele hidden sau system.
  if ( !findfirst ( "*.*", &w, FA_DIREC + FA_HIDDEN + FA_SYSTEM ) )
        do      {
```


Scanned with OKEN Scanner

```
            if ( (w.ff_attrib & FA_DIREC) == 0 )
                continue ;
            if ( strcmp(".",w.ff_name) && strcmp("..",w.ff_name) ) {
                    // nu este vorba de "." sau ".."
                chdir(w.ff_name) ;
                parcurgereCompleta() ;      // apelare recursivă !
                }
            }       while ( !findnext(&w) ) ;
 chdir("..") ;
}
```

Problema 24.1

Să se determine şi să se afişeze numerele logice pentru dispozitivele de I/E standard DOS.

```
#include <stdio.h>
#include <conio.h>
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
        clrscr() ;
        printf ( "STDIN  : %d\n", fileno ( stdin ) ) ;
        printf ( "STDOUT : %d\n", fileno ( stdout ) ) ;
        printf ( "STDERR : %d\n", fileno ( stderr ) ) ;
        printf ( "STDAUX : %d\n", fileno ( stdaux ) ) ;
        printf ( "STDPRN : %d\n", fileno ( stdprn ) ) ;
        printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
        getch() ;
        return 0 ;
}
```

Problema 24.2

Să se scrie un program care modifică data şi ora unui fişier în valorile curente (se va lucra cu funcţiile **gettime()** şi **getdate()**).

Se va compara programul propriu cu **TOUCH**, din kitul Turbo C++.

```
#include <stdio.h>
```


Scanned with OKEN Scanner

```
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <io.h>
#include <string.h>
/*-------------------------------------------------------------------------------------------*/
int main ( int argc , char *argv[] )
/*
        Apelare : 24.2 file_name [/H | /?]
*/
{
 struct date dt ;
 struct time tp ;
 struct ftime ft ;
 FILE *f ;
 char comanda[80] ;
 clrscr() ;
 if ( argc==1 ||   ( argc>1 && (!stricmp(argv[1],"/h") || !strcmp(argv[1],"/?") ) ) ) {
        printf("\nFolosire program : 24_2 filename [/H | /?] ") ;
        exit(1) ;
        }
 sprintf(comanda,"DIR %s ",argv[1]) ;
 printf("\nInainte de modificare :\n") ;
 if ( system(comanda) )
        perror("\n\aLansare DIR   ") ;
 if ( ( f = fopen(argv[1],"r") ) == NULL )          {
        perror("Deschidere fisier ") ;
        exit(2) ;
        }
 getdate(&dt) ;
 gettime(&tp) ;
 ft.ft_year  = dt.da_year - 1980 ;
 ft.ft_month = dt.da_mon ;
 ft.ft_day   = dt.da_day ;
 ft.ft_hour  = tp.ti_hour ;
 ft.ft_min   = tp.ti_min ;
 ft.ft_tsec  = tp.ti_sec ;
 if ( setftime(fileno(f),&ft) )
        perror("\n\aEroare fixare data/ora ") ;
 fclose(f) ;
 printf("\nDupa modificare :\n") ;
```


Scanned with OKEN Scanner

```
 if ( system(comanda) )
        perror("\n\aLansare DIR   ") ;
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 24.4

Să se scrie un program care converteşte *tab*-urile dintr-un fişier de intrare într-un număr predefinit de spaţii.

Fişierul de ieşire va avea aceleaşi nume, extensie, atribute, dată şi oră ca fişierul de intrare, acestuia din urmă schimbîndu-i-se extensia în **.BAK** (un eventual fişier cu acelaşi nume şi extensia **.BAK** va trebui să dispară - atenţie la posibilitatea de a fi *read-only*).

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <io.h>
#include <process.h>
#include <dos.h>
#include <sys\stat.h>
#define NUMAR_SPATII 2
/*-------------------------------------------------------------------------------------*/
int main ( void )
{
 char numeFisier[80] ;
 char numeBak[80] ;
 int k , handleFis , c , nrs ;
 struct ftime dt ;
 FILE *f , *g ;
 clrscr() ;
 printf("Numele fisierului de intrare : \n") ;
 gets(numeFisier) ;
 if (!strlen(numeFisier))
        exit(1) ;
 if ( access ( numeFisier, 0 ) )    {
        printf("\nNu exista fisierul %s !",numeFisier) ;
        exit(1) ;
```


Scanned with OKEN Scanner

```
        }
strcpy(numeBak,numeFisier) ;  // se creează numele fişierului .BAK
for (k=0; k<strlen(numeFisier); k++)
        if ( numeFisier[k]=='.' || numeFisier[k]=='\0' )
                break ;
numeBak[k] = '\0' ;
strcat(numeBak,".BAK") ;
if ( access ( numeBak, 0 ) == 0 )  {       // se şterge un eventual fişier .BAK existent
        chmod ( numeBak, S_IREAD + S_IWRITE ) ;
        unlink ( numeBak ) ;
        }
rename ( numeFisier, numeBak ) ;
if ( ( f = fopen ( numeBak, "rb" ) ) == NULL )              {
        perror("\nEroare in deschiderea fisierului de intrare ") ;
        exit(1) ;
        }
handleFis = fileno(f) ;
getftime(handleFis,&dt) ;
if ( ( g = fopen ( numeFisier, "wb" ) ) == NULL )          {
        perror("\nEroare in deschiderea fisierului de iesire ") ;
        exit(1) ;
        }
while ( !feof(f) )          {
        c = getc(f) ;
        if ( c != '\t' )
                putc(c,g) ;
        else       {
                for ( nrs=0; nrs<NUMAR_SPATII; nrs++)
                        putc(' ',g) ;
                }
        }
 fcloseall() ;
/*
        Se redeschide fişierul nou creat numai în citire (altfel, sistemul de operare ar
fixa - transparent pentru utilizatori - data şi ora ca cele din momentul închiderii).
*/
 if ( ( g = fopen ( numeFisier, "rb" ) ) == NULL )                    {
        perror("\nEroare redeschidere fisier ") ;
        exit(1) ;
        }
 handleFis = fileno(g) ;
```


Scanned with OKEN Scanner

```
 setftime(handleFis,&dt) ;
 fclose(g) ;
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 25.1

Să se scrie un program care citeşte un fişier text şi creează un fişier de ieşire în care apar liniile fişierului de intrare numerotate.

Pentru simplificare, se caută mai întîi numele fişierului de intrare în linia de comandă, iar fişierul de ieşire are numele celui de intrare şi extensia .NUM.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <dir.h>
/*-------------------------------------------------------------------------------------------*/
int main ( int argc , char *argv[] )
{
 int nrLinie = 0 ;
 char linieText[85] ;
 FILE *f , *g ;
 char numeFisIn[80] ;
 char numeFisIes[80] ;
 char unDisc[3] , numeDirec[66] , numeFis[9] , numeExt[5] ;
 clrscr() ;
 if ( argc>1 )
        strcpy(numeFisIn,argv[1]) ;
 else   {
        printf("\nNumele fisierului de intrare : ") ;
        gets(numeFisIn) ;
        }
 f = fopen(numeFisIn,"r") ;
 if ( f == NULL )       {
        perror("\n\aEroare deschidere fisier intrare ") ;
        exit(1) ;
        }
```


Scanned with OKEN Scanner

```
 fnsplit ( numeFisIn , unDisc , numeDirec , numeFis , numeExt ) ;
 fnmerge ( numeFisIes , unDisc , numeDirec , numeFis , ".NUM" ) ;
 g = fopen(numeFisIes,"w") ;
 if ( g == NULL )        {
     -    perror("\n\aEroare deschidere fisier iesire ") ;
          exit(2) ;
          }
 while ( fgets ( linieText , 80 , f ) != NULL )
          fprintf(g,"%4d %s",nrLinie++,linieText) ;
 fclose(f) ;
 fclose(g) ;
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch();
 return 0 ;
}
```

Problema 25.4

Să se scrie un program care concatenează într-un fişier TOTAL.LST toate fişierele antet ale Turbo C++ (directoarele \TC\INCLUDE şi \TC\INCLUDE\SYS).

Fiecare fişier va începe de la o pagină nouă, cu numele scris în centrul primei linii; paginile vor fi numerotate pe ultima linie, cu numărul de pagină scris tot în centrul liniei.

Se presupune că lungimea unei pagini este de 66 de linii; o linie are 80 de coloane.

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <process.h>
#include <dir.h>
#include <string.h>
/*-------------------------------------------------------------------------------*/
int main ( void )
{
 FILE *f, *g ;
 char linie[100] ;
 char numeFisier[80] ;
 char mesajEroare[100] ;
 int i ;
```


Scanned with OKEN Scanner

```
int numarPagina = 1 ;
int numarLinii ;
struct ffblk w ;
clrscr() ;
if ( ( f = fopen ( "TOTAL.LST", "w" ) ) == NULL )      {
        printf("\nNu poate fi deschis fisierul de iesire !") ;
        exit(1) ;
        }
if ( !findfirst ( "E:\\TC\\INCLUDE\\*.H", &w, FA_ARCH ) ) {
                        // se indică unitatea de disc şi directorul corespunzătoare
        while(1)        {
                strcpy(numeFisier,"E:\\TC\\INCLUDE\\") ;
                strcat(numeFisier,w.ff_name) ;
                if ( ( g = fopen ( numeFisier, "r" ) ) == NULL )   {
                        sprintf(mesajEroare,"\nNu poate fi deschis %s!",
                                numeFisier) ;
                        perror(mesajEroare) ;
                        exit(1) ;
                        }
                for ( i=0; i < ( 80 - strlen(w.ff_name) ) / 2; i++ )
                        putc(' ',f) ;
                // Se scrie numele fişierului în centrul primei linii.
                fprintf(f,"%s\n\n\n\n",w.ff_name) ;
                numarLinii = 5 ;
                while ( !feof(g) )          {
                        if (fgets(linie,100,g)==NULL)
                                break ;
                        fputs(linie,f) ;
                        numarLinii++ ;
                        if ( numarLinii == 65 )  {
                                fprintf(f,"\n") ;
                                for (i=0; i<38; i++)
                                        putc(' ',f) ;
                                fprintf ( f, "-%d-\n\f", numarPagina ) ;
                                // Se scrie numărul paginii în centrul ultimei linii.
                                numarLinii = 1 ;
                                ++numarPagina ;
                                }
                        }
                fclose(g) ;
                if (numarLinii!=1)      {
```


Scanned with OKEN Scanner

```
                    // Se completează eventual ultima pagină cu linii goale.
                    while ( numarLinii <= 65 )        {
                            putc('\n',f) ;
                            numarLinii++ ;
                            }
                    for (i=0; i<38; i++)
                            putc(' ',f) ;
                    fprintf(f,"-%d-\n\f",numarPagina) ;
                    ++numarPagina ;
                    }
            if (findnext(&w))          // nu mai există fişiere ca cele căutate
                    break ;
            }
        }
            // Adăugaţi şi codul necesar pentru \INCLUDE\SYS.
 printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
 getch() ;
 return 0 ;
}
```

Problema 26.1

Să scrie un program de eliminare dintr-un fişier a unui virus cu următoarele caracteristici :

- atacă fişierele .COM şi .EXE, mai puţin COMMAND.COM ;

- marchează fişierul virusat prin cuvîntul **0x4456** după 26 de octeţi de la începutul fişierului ;

- adaugă la sfîrşitul iniţial 778 octeţi, din care primii 746 reprezintă virusul efectiv iar următorii 28 începutul fişierului nevirusat (ultimii 4 octeţi nu au vreo semnificaţie, apar numai datorită alocării memoriei în paragrafe de cîte 16 octeţi);

Fişierul original este CRC.COM, fişierul virusat este CRC.VIR iar cei 746 octeţi menţionaţi se află în fişierul COD.VIR. După (tentativa de) eliminare, se vor compara din program fişierul original cu cel (fost ?) virusat.

Se va avea în vedere posibilitatea creşterii lungimii fişierului virusat cu 5, 10, 15 sau 20 de octeţi prin adăugarea unor sume de control de către programe

Scanned with OKEN Scanner

(antivirus) ce nu cunosc virusul respectiv (secvențele respective - devenite oricum irelevante - vor fi și ele eliminate !).

În urma devirusării nu trebuie să se modifice atributele sau data și ora pentru fișierul prelucrat !

```
#include <stdio.h>
#include <conio.h>
#include <io.h>
#include <dir.h>
#include <dos.h>
#include <stdlib.h>
#include <mem.h>
#include <string.h>
#include <sys\stat.h>
char secventaProgram[746] ;
char corpVirus[746] ;
char antetFisier[28] ;
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
  char numeFisier[80] ;
  FILE *f, *g ;
  long lungimeFisier ;
  int c , k , handleFisier ;
  int atributeFisier ;
  struct ftime data ;
  clrscr() ;
  printf("Numele fisierului de verificat :\n") ;
  gets(numeFisier) ;
  if ( !strlen(numeFisier) )
        exit(1) ;
  if ( access(numeFisier,0) )       {
        printf("\nNu exista fisierul %s !",numeFisier) ;
        exit(1) ;
        }
  if ( ( f = fopen(numeFisier,"rb") ) == NULL )   {
        printf("\nNu poate fi deschis fisierul %s !",numeFisier) ;
        exit(1) ;
        }
  fseek(f,26L,SEEK_SET) ;
  c = getw(f) ;
```


Scanned with OKEN Scanner

```
if ( c != 0x4456 )        {
        // atentie la incersarea octeților în cuvînt ( dacă se citesc pe rînd 2 octeți ) !
        printf("\n%s nu este virusat !",numeFisier) ;
        exit(1) ;
        }
fclose(f) ;
                // Se memorează atributele fișierului după care se șterg (toate).
atributeFisier = _chmod ( numeFisier, 0 ) ;
if ( _chmod ( numeFisier , 1, 0) )        {
        printf("\n%s nu poate fi facut R/W !",numeFisier) ;
        exit(1) ;
        }
                // Se redeschide fișierul vizat în citire cu actualizare.
if ( ( f = fopen(numeFisier,"r+b") ) == NULL ) {
        printf("\nNu poate fi deschis %s !",numeFisier) ;
        exit(1) ;
        }
handleFisier = fileno(f) ;
lungimeFisier = filelength(handleFisier) ;
getftime(handleFisier,&data) ;
if ( ( g = fopen("COD.VIR","rb") ) == NULL ) {
        printf("\nNu poate fi deschis COD.VIR !") ;
        exit(1) ;
        }
fread(corpVirus,1,746,g) ;
fclose(g) ;
                // Se caută (în 5 locuri) corpul virusului în fișierul vizat.
for ( k=4; k>=0; k-- )  {
        if ( fseek ( f, -778L-k*5, SEEK_END) ) {
                printf("\nEroare pozitionare in fisier !") ;
                exit(1) ;
                }
        fread ( secventaProgram, 1, 746, f ) ;
        if ( !memcmp ( secventaProgram, corpVirus, 746 ) )        {
                fseek ( f, -32L - k * 5 , SEEK_END ) ;
                fread ( antetFisier, 1, 28, f ) ;              // citire antet inițial
                rewind(f) ;
                fwrite ( antetFisier, 1, 28, f ) ;             // rescriere antet inițial
                chsize ( handleFisier, lungimeFisier - 778L - k * 5 ) ;
                                // s-a trunchiat zona corespunzătoare virusului
                printf("\nFisierul %s a fost virusat !",numeFisier) ;
```


Scanned with OKEN Scanner

```
                    break ;
                    }
            }
    fclose(f) ;
                    // Se redeschide fişierul numai în citire pentru a fixa data-ora.
    if ( ( f = fopen(numeFisier,"rb") ) == NULL )    {
            printf("\nNu poate fi redeschîs fisierul !") ;
            exit(1) ;
            }
    handleFisier = fileno(f) ;
    setftime(handleFisier,&data) ;
    fclose(f) ;
                    // Se refac atributele iniţiale.
    _chmod ( numeFisier, 1, atributeFisier ) ;
    printf("\n\n\t\tApasati orice tasta pentru terminare...") ;
    getch() ;
    return 0 ;
}
```

Problema 26.2

Dezvoltaţi programul **26.1** pentru eliminarea virusului prezentat de pe o unitate de disc indicată de la tastatură; se vor afişa numele complete ale tuturor fişierelor găsite virusate; în final se va reveni la unitatea de disc / directorul de plecare.

```
#include <stdio.h>
#include <conio.h>
#include <io.h>
#include <dir.h>
#include <dos.h>
#include <stdlib.h>
#include <mem.h>
#include <string.h>
#include <sys\stat.h>
#include <ctype.h>
char secventaProgram[746] ;
char corpVirus[746] ;
char antetFisier[28] ;
void parcurgereCompleta(void) ;
```


Scanned with OKEN Scanner

```
void dezinfectFisiere(void) ;
int  eliminareVirus(char *nume) ;
/*-------------------------------------------------------------------------------------------*/
int main ( void )
{
 int c , d ;
 FILE *g ;
 char directorInitial[80] ;
 clrscr() ;
 if ( ( g = fopen("COD.VIR","rb") ) == NULL)  {
        printf("\aNu poate fi deschis COD.VIR !") ;
        exit(1) ;
        }
 fread(corpVirus,1,746,g) ;
 fclose(g) ;
 d = getdisk() ;
 getcwd(directorInitial,80) ;
 printf("\nUnitatea de verificat : ") ;
 do     {
        c = getch() ;
        if ( c == 0x1b )  // ESC
                exit(1) ;
        }                       while ( c<'A' || c>'Z' && c<'a' || c>'z' ) ;
 printf("%c\n",c) ;
 if ( tolower(c) - 'a' >= setdisk(getdisk()) )          {
        printf("\n\aUnitate eronata !") ;
        exit(1) ;
        }
 setdisk(tolower(c)-'a') ;
 chdir("\\") ;
 parcurgereCompleta() ;
 setdisk(d) ;
 chdir(directorInitial) ;
 printf("\nProgramul s-a încheiat.") ;
 return 0 ;
}
/*-------------------------------------------------------------------------------------------*/
/*
        Funcţia eliminareVirus face eliminarea virusului vizat dintr-un fişier.
        Parametri:
                - nume: numele fişierul  în care se caută;
```


Scanned with OKEN Scanner

```
        Returnează:
                - int: 0 pentru terminare cu succes (fişierul nu este virusat
                        sau a fost dezinfectat), altfel -1;
*/
int eliminareVirus ( char *nume )
{
 char direc[80] ;
 FILE *f ;
 long lungimeFisier ;
 int accesFisier , handleFisier , c , k ;
 struct ftime data ;
 if ( ( f=fopen(nume,"rb") ) == NULL )
        return -1 ;
 fseek ( f, 26L, SEEK_SET ) ;
 c = getw(f) ;
 fclose(f) ;
 if ( c != 0x4456 )
        return 0 ;
 accesFisier = access ( nume, 06 ) ;
 if ( accesFisier )
        if ( chmod ( nume, S_IREAD + S_IWRITE ) )
                return -1 ;
 if ( ( f = fopen ( nume, "r+b" ) ) == NULL )
        return -1 ;
 handleFisier = fileno(f) ;
 lungimeFisier = filelength(handleFisier) ;
 getftime(handleFisier,&data) ;
 for (k=4; k>=0; k--)                {
        if (fseek(f,-778L-k*5,SEEK_END))        {
                fclose(f) ;
                return -1 ;
                }
        fread(secventaProgram,1,746,f) ;
        if (!memcmp(secventaProgram,corpVirus,746))   {
                fseek(f,-32L-k*5,SEEK_END) ;
                fread(antetFisier,1,28,f) ;
                rewind(f) ;
                fwrite(antetFisier,1,28,f) ;
                chsize(handleFisier,lungimeFisier-778L-k*5) ;
                getcwd(direc,80) ;
                if (strlen(direc)==3)       \\ să nu apară '\' dublat pentru rădăcină
```


Scanned with OKEN Scanner

```
                printf("%s%s a fost devirusat .\n",direc,nume) ;
            else
                printf("%s\\%s a fost devirusat .\n",direc,nume) ;
            break ;
            }
        }
    fclose(f) ;
    if ( ( f = fopen ( nume, "rb" ) ) == NULL )
        return -1 ;
    handleFisier = fileno(f) ;
    setftime(handleFisier,&data) ;
    fclose(f) ;
    if ( accesFisier )
        chmod(nume,S_IREAD) ;
    return 0 ;
}
/*-------------------------------------------------------------------------------*/
/*
```

Funcția **parcurgereCompleta** asigură trecerea prin toate directoarele (după stabilirea unității de disc și trecerea în rădăcină).

Parametri:

- *void*;

Returnează:

- *void*;

```
*/
void parcurgereCompleta ( void )
{
    struct ffblk w ;
    dezinfectFisiere() ;
    if ( !findfirst ( "*.*", &w, FA_DIREC + FA_HIDDEN + FA_SYSTEM ) )
        do      {
            if ( (w.ff_attrib & FA_DIREC) == 0 )    // este un director
                continue ;
            if ( stremp(".",w.ff_name) && stremp("..",w.ff_name) ) {
                chdir(w.ff_name) ;
                parcurgereCompleta() ;
                }
            }       while ( !findnext(&w) ) ;
    chdir("..") ;
}
/*-------------------------------------------------------------------------------*/
```


Scanned with OKEN Scanner

```
/*
        Funcţia dezinfectFisiere caută toate fişierele cu extensia .COM sau .EXE,
după care apeleaza funcţia de eliminare a virusului.
        Excepţie: COMMAND.COM.
        Parametri:
                - void;
        Returnează:
                - void;
*/
void dezinfectFisiere ( void )
{
 struct ffblk q ;
 char numeDirector[80] ;
 char temp[90] ;
 if ( !findfirst ( "*.COM", &q,
                FA_ARCH + FA_RDONLY + FA_HIDDEN + FA_SYSTEM ) )
  while ( 1 )    {
        if ( stricmp("COMMAND.COM",q.ff_name) )
                if (eliminareVirus(q.ff_name))    {
                    sprintf(temp,"\n%s\\%s",getcwd(numeDirector,80),q.ff_name) ;
                    perror(temp) ;
                    }
        if (findnext(&q))                          break ;
        }
 if ( !findfirst ( "*.EXE", &q,
                FA_ARCH+FA_RDONLY+FA_HIDDEN+FA_SYSTEM))
   while(1)      {
        if (eliminareVirus(q.ff_name))    {
                sprintf(temp,"\n%s\\%s",getcwd(numeDirector,80),q.ff_name) ;
                perror(temp) ;
                }
        if (findnext(&q))                          break ;
        }
}
```

Problema 27.3

Să se scrie un program care afişează toate fişierele de pe o unitate de disc ce au ca dată a creării / ultimei modificări o valoare introdusă de la tastatură. Se va

Scanned with OKEN Scanner

preciza dacă este vorba de un director; pentru toate fişierele găsite se vor afişa numele complet şi ora corespunzătoare.

```
#include <stdio.h>
#include <conio.h>
#include <io.h>
#include <dir.h>
#include <dos.h>
#include <stdlib.h>
#include <mem.h>
#include <string.h>
#include <sys\stat.h>
#include <ctype.h>
#include <time.h>
typedef struct {
        int zi ;
        char luna[4] ;
        int an ;
        } DATAFIS ;
int  compararеDate ( DATAFIS valoareData , struct stat fs ) ;
void parcurgereCompleta ( DATAFIS valoareData ) ;
void cautaData ( DATAFIS valoareData ) ;
/*-------------------------------------------------------------------------------------*/
int main ( int argc , char *argv[] )
{
 int unParcursa , unInitiala ;
 char direcInitial[80] ;
 char dataFisier[80] ;
 DATAFIS df ;
 int bisect ;
 int lungimiLuni[] = { 31 , 28 , 31 , 30 , 31 , 30 , 31 , 31 , 30 , 31 , 30 , 31 } ;
 char *numeLuni[] = { "Jan" , "Feb" , "Mar" , "Apr" , "May" , "Jun" ,
                      "Jul" , "Aug" , "Sep" , "Oct" , "Nov" , "Dec" } ;
 int zi , luna , an ;
 clrscr() ;
 if ( argc>1 ) // se caută mai întîi data vizată ca parametru în linia de comandă
        strcpy(dataFisier,argv[1]) ;
 else   {
        printf("\nData dorita ( zz/ll/|aa|aa ) : ") ;
        gets(dataFisier) ;
        }
```


Scanned with OKEN Scanner

```
 if ( sscanf ( dataFisier, "%2d/%2d/%4d", &zi, &luna, &an) != 3 )        {
        printf("\n\n\aFormatul corect este  dd/ll/[aa]aa !") ;
        exit(1) ;
        }
//   Se poate adăuga şi formatul dd-mmm-[aa]aa pentru data dorită - vezi problema
// 21.1.
 luna-- ;
 if (an>=80 && an<=99)
        an += 1900 ;
 else if (an>=0 && an<80)
        an += 2000 ;
 if (an<1980 || an>2079)          {
        printf("\n\n\aDOS cunoaste numai intervalul 1980-2079 !") ;
        exit(2) ;
        }
 if ( luna<0 || luna>=12 )        {
        printf("\n\n\aExista luna numarul %d ?! ",luna) ;
        exit(3) ;
        }
 bisect = an%4==0 && an%100!=0 || an%400==0 ;
 lungimiLuni[1] += bisect ;
 if ( zi<1 || zi>lungimiLuni[luna] )        {
        printf("\n\n\aAre luna %s ziua cu numarul %d in anul %d ?!",
                        numeLuni[luna] , zi , an ) ;
        exit(4) ;
        }
 df.zi = zi ;
 strcpy(df.luna,numeLuni[luna]) ;
 df.an = an ;
 unInitiala = getdisk() ;
 getcwd(direcInitial,80) ;
 printf("\nUnitatea de verificat : ") ;
 do     {
        unParcursa = getch() ;
        if (unParcursa==0x1b)                   exit(1) ;
        } while ( unParcursa<'A' || unParcursa>'Z' && unParcursa<'a' ||
                unParcursa>'z' ) ;
 printf("%c\n",unParcursa) ;
 if (tolower(unParcursa)-'a'>=setdisk(getdisk()))  {
        printf("\n\aUnitate eronata !") ;
        exit(1) ;
```


Scanned with OKEN Scanner

```
        }
setdisk(tolower(unParcursa)-'a') ;
chdir("\\") ;
parcurgereCompleta ( df ) ;
setdisk(unInitiala) ;
chdir(direcInitial) ;
printf("\nProgramul s-a încheiat !") ;
return 0 ;
}
/*-------------------------------------------------------------------------------------------*/
/*
        Funcția parcurgereCompleta asigură trecerea prin toate directoarele (după
stabilirea unității de disc și trecerea în rădăcină).
        Parametri:
                - valoareData: data calendaristică vizată;
        Returnează:
                - void;
*/
void parcurgereCompleta ( DATAFIS.valoareData )
{
 struct ffblk w ;
 cautaData ( valoareData ) ;
 if ( !findfirst ( "*.*", &w, FA_DIREC+FA_HIDDEN+FA_SYSTEM ) )
        do      {
                if ( (w.ff_attrib & FA_DIREC) == 0 )            continue ;
                if (strcmp(".",w.ff_name) && strcmp("..",w.ff_name))  {
                        chdir(w.ff_name) ;
                        parcurgereCompleta(valoareData) ;
                        }
                }       while ( !findnext(&w) ) ;
 chdir("..") ;
}
/*-------------------------------------------------------------------------------------------*/
/*
        Funcția cautaData caută toate fișierele din directorul curent ce au ca dată
valoarea fixată în structura VAL.
        Parametri:
                - valoareData: data vizată;
        Returnează:
                - void;
*/
```


Scanned with OKEN Scanner

```
void cautaData ( DATAFIS valoareData )
{
 struct ffblk q ;
 struct stat w ;
 char numeDir[66] ;
 char oraFis[10] ;
 getcwd(numeDir,66) ;
 if ( !findfirst ( "*.*", &q,
      FA_DIREC+FA_ARCH+FA_RDONLY+FA_HIDDEN+FA_SYSTEM ) )
  do    {
        stat ( q.ff_name , &w ) ;
        if ( compararеDate ( valoareData , w ) )  {
                printf("%s - %s",numeDir,q.ff_name) ;
                strncpy ( oraFis , ctime(&w.st_atime) + 11 , 8 ) ;
                oraFis[8] = '\0' ;
                printf("  : %s ",oraFis ) ;
                if ( (w.st_mode & S_IFDIR) != 0 )
                                printf("   DIRECTOR") ;
                printf("\n") ;
                }
        }
        while ( !findnext(&q) ) ;
}
/*-------------------------------------------------------------------------------------------*/
/*
```

Funcția **compararеDate** face compararea între două date calendaristice, returnînd 1 pentru coincidență, altfel 0.

Parametri:

- *valoareData*: prima dată calendaristică;
- *fs*: a doua prima dată calendaristică;

Returnează:

- *int*: 1 pentru coincidență, altfel 0;

```
*/
int compararеDate ( DATAFIS valoareData , struct stat fs )
{
 char *temp ;
 int ziua , anul ;
 char luna[4] ;
 temp = ctime ( &fs.st_atime ) ;
 sscanf ( temp+3, "%s %d", luna, &ziua) ;
 sscanf ( temp+19, "%d", &anul ) ;
```


Scanned with OKEN Scanner

```
 if ( valoareData.zi==ziua && !stricmp(valoareData.luna,luna) &&
       valoareData.an==anul )
          return 1 ;
 else
          return 0 ;
}
```

Problema 28.2

Reluați problema **26.2**, folosind de această dată EXCLUSIV funcțiile de lucru cu fișiere prin intermediul *handle*.

Se va căuta următoarea "semnătură" a virusului (15 octeți, în hexa) :

53	8a	d8	0a	c4	75	02	24
7f	9c	2e	ff	1e	0f	03	

aflată cu 547 octeți înainte de sfîrșitul celor 778 de octeți reprezentînd corpul virusului.

> **Notă** : Virusul pe care îl "decupați" este denumit **[BIT7]** (pe plan local), sau **[Mannequin]** (SCAN) sau **[Jassy-778]** (AIDSTEST).

Se prezintă numai varianta adaptată a funcției de eliminare a virusului :

```
int eliminareVirus2 ( char *nume )
{
 char numeDirector[80] ;
 int handleFisier ;
 long lungimeFisier ;
 int atributeFisier , k ;
 char c[2] ;
 struct ftime data ;
 if ( ( handleFisier = open(nume,O_RDONLY+O_BINARY) ) == -1 )
          return -1 ;
 lseek ( handleFisier, 26L, SEEK_SET ) ;
 read ( handleFisier, c, 2 ) ;
 close ( handleFisier ) ;
 if ( c[0]!=0x56 || c[1]!=0x44 )
          return 0 ;
```


Scanned with OKEN Scanner

```
  atributeFisier = _chmod ( nume, 0 ) ;
  _chmod ( nume, 1, 0 ) ;
  if ( ( handleFisier = open ( nume, O_RDWR + O_BINARY ) ) == -1 )
        return -1 ;
  lungimeFisier = filelength(handleFisier) ;
  getftime(handleFisier,&data) ;
  for (k=4; k>=0; k--)              {
        if ( lseek ( handleFisier, -547L-k*5, SEEK_END ) == -1L )       {
              close(handleFisier) ;
              return -1 ;
              }
        read ( handleFisier, secventaProgram, 15 ) ;
// semnaturaVirus[] este o variabilă globală, inițializată în main() cu valorile
// corespunzătoare.
        if ( !memcmp ( secventaProgram, semnaturaVirus, 15 ) ) {
              lseek ( handleFisier, -32L-k*5, SEEK_END ) ;
              read ( handleFisier, antetFisier, 15 ) ;
              lseek ( handleFisier, 0L, SEEK_SET ) ;
              write ( handleFisier, antetFisier, 28 ) ;
              chsize ( handleFisier, lungimeFisier-778L-k*5 ) ;
              getcwd(numeDirector,80) ;
              if (strlen(numeDirector)==3)
                    printf("%s%s a fost devirusat .\n",numeDirector,nume) ;
              else
                    printf("%s\\%s a fost devirusat .\n",numeDirector,nume) ;
              break ;
              }
        }
  setftime(handleFisier,&data) ;
  close(handleFisier) ;
  _chmod ( nume, 1, atributeFisier ) ;
  return 0 ;
}
```


Scanned with OKEN Scanner

B. Probleme suplimentare

B1) Să se scrie un program prin care se citeşte un text din intrarea standard şi se afişează linia cea mai lungă.

```
#include <stdio.h>
#include <conio.h>
#define LMAX 100
int getlinie(char s[],int lim);
void copy(char s1[],char s2[]);
void main(void)
{
int lc;         /* lungimea liniei curente */
int max = 0 ;     /* lungimea maxima gasita pina acum */
char linie[LMAX]; /* linia curenta introdusa */
char salv[LMAX];  /* linia cea mai lunga, salvata */
clrscr();
printf("\nIntroduceti liniile textului:\n\n");
while(( lc = getlinie(linie,LMAX)) > 0)
        if(lc > max)  {
                max = lc;
                copy(linie,salv);
                }
if(max > 0)       /* s-a citit cel putin o linie */
        printf("\nLinia cea mai lunga din text este:\n%s\n",salv);
getch();
}

int getlinie(char s[],int lim)
{
int c,i;
for(i = 0; i < lim && (c = getchar()) != '\n' && c != EOF; ++i)
        s[i] = c;
if( c == '\n')  {
        s[i] = c;
        ++i;
        }
s[i] = '\0';
return(i);
```


Scanned with OKEN Scanner

```
}

void copy(char s1[],char s2[])
{
int i = 0;
while((s2[i] = s1[i]) != '\0')
        ++i;
}
```

B2) Să se scrie un program prin care se calculează frecvența de apariție a cuvintelor unui text citit din intrarea standard. Cuvintele şi numărul de apariții al fiecăruia vor fi afişate în ordinea citirii lor,deci nu în ordine alfabetică.

Pentru rezolvarea problemei se impune folosirea unei liste dublu înlănțuite,dar nu se impune crearea listei prin inserție, pentru că nu se impune aranjarea în ordine alfabetica a elementelor listei.

Se vor scrie funcții pentru a realiza următoarele operații:

- încărcarea unui nod (element) al listei cu informații utile,adică cu cîte un cuvint;
- eliberarea unui nod al listei dublu înlănțuite;
- căutarea unui nod ce conține o anumită informație;
- adăugarea unui nod în lista dublu înlănțuită;
- ştergerea ultimului nod;
- ştergerea listei.

```
#include <stdio.h>
#include <string.h>
#include <alloc.h>
#include <conio.h>
#define MAXCUV 50
typedef struct tnod  {
        char *cuvint;
        int frecventa;
        struct tnod *prec;
        struct tnod *urm;
        } TNOD;
int incnod(TNOD *);   /*incarca nod */
void elibnod(TNOD *); /*elibereaza nod */
TNOD *dadauga(void);  /*dubla - adauga nod in lista */
TNOD *dcncs(char *);  /*dubla - cauta nodul ce contine sirul */
void dsun(void);      /*dubla - sterge ultimul nod */
```


Scanned with OKEN Scanner

```
void dsterglist(void):/*dubla - sterge lista */

TNOD *prim,*ultim;

void main(void)  /* citeste un text si afiseaza frecventa de aparitie a
                                                cuvintelor din text */
{
TNOD *p,*q;
prim = ultim = NULL;
clrscr();
printf("\n Introduceti cuvintele unui text:\n");
while((p = dadauga()) != NULL)  {
        q = denes(p->cuvint);
/* se cauta cuvintul curent adaugat la lista dublu inlantuita;q este pointer spre nodul
in care a fost el adaugat:daca q=p,atunci cuvintul curent nu era in lista;daca q!=p,
atunci cuvintul era in lista*/
        if(q != p) {
                q->frecventa++;   /* incrementez contorul cuvintului gasit*/
                        dsun();      /* sterg ultimul nod,caci cuvintul se afla
                                                        deja in lista */
                }
        }
printf("\n Ati introdus urmatoarele cuvinte:\n\n");
p = prim;
while(p != NULL) {
        printf("%-20s %4d\n",p->cuvint,p->frecventa);
        p = p->urm;
        }
getch();
dsterglist();
}

TNOD *dadauga(void) /* adauga un nod in lista dublu inlantuita */
{
TNOD *p;
int n;
n = sizeof(TNOD);
if(((p = (TNOD *)malloc(n)) != NULL) && (incnod(p) == 1))  {
        if(prim == NULL)  {   /* lista este vida */
                prim = ultim = p;
                p->prec = p->urm = NULL;
```


Scanned with OKEN Scanner

```
                return p;
                }
        ultim->urm = p;
        p->prec = ultim;
        p->urm = NULL;
        ultim = p;
        return p;
        }
if(p == NULL) {
        printf("\n Memorie insuficienta \n");
        return p;
        }
elibnod(p);
return(NULL);
}

TNOD *dcncs(char *p)   /*cauta nodul ce contine sirul p */
{
TNOD *q;
q = prim;
while(q != NULL)
        if(strcmp(p,q->cuvint) == 0)
                return q;
        else
                q = q->urm;
        return NULL;
}

void dsun()         /* sterge ultimul nod din lista */
{
if(ultim == NULL)
        return;         /* lista este vida */
if(ultim->prec == NULL)  {  /*lista are un singur nod */
        elibnod(ultim);
        prim = ultim = NULL;
        return;
        }
ultim = ultim->prec; /*penultimul nod devin ultimul */
elibnod(ultim->urm); /*eliberez ultimul nod */
ultim->urm = NULL;   /*actualizarea ultimului nod*/
}
```


Scanned with OKEN Scanner

```
void dsterglist()    /*sterge o lista dublu inlantuita */
{
TNOD *p,*p1;
p = prim;
while(p != NULL) {
        p1 = p->urm;
        elibnod(p);
        p = p1;
        }
prim = ultim = NULL;
}

void elibnod(TNOD *p)  /* elibereaza zona de memorie pointata de p*/
{
free(p->cuvint);       /*elibereaza zona ocupata de cuvint */
free(p);
}

int incnod(TNOD *p)    /*incarca nodul*/
{
int i,c;
char cuvcrt[MAXCUV + 1];
char *q;
p->cuvint = NULL;
while((c = getchar()) != EOF)
        if(c >= 'A' && c <= 'Z' || c >= 'a' && c <= 'z')
                break;  /* sar peste caractere care nu-s litere */
        if(c == EOF)
                return -1;  /* sfirsitul textului */
for(i = 0; i <= MAXCUV && (c >= 'A' && c <= 'Z' || c >= 'a' && c <= 'z');i++) {
        cuvcrt[i] = c; /*pastreaza caracterele cuvintului */
        c = getchar();
        }
if(c == EOF)
        return -1;     /* sfirsitul textului */
if(i > MAXCUV)   /* cuvint cu prea multe caractere*/
        return 0;
q = (char *)malloc(i+1); /*rezervare memorie pentru cuvint*/
cuvcrt[i] = '\0';
strcpy(q,cuvcrt);
```


Scanned with OKEN Scanner

```
p->cuvint = q;    /* se incarca in nod pointerul unde este cuvintul citit*/
p->frecventa = 1;
return 1;
}
```

B3) Să se scrie un program prin care se realizează citirea şi apoi afişarea membrilor unei structuri de tip dată calendaristică.

```
#include <stdio.h>
#include <conio.h>

typedef struct data_calendar {
        int zi;
        char nume_luna[12];
        int an;
        }DC;

DC data_nas,data_ang;

void main(void)
{
clrscr();
printf("\n Specificati data nasterii:\n\n");
printf(" ziua: ");
scanf("%d",&data_nas.zi);
printf(" luna: ");
scanf("%s",data_nas.nume_luna);
printf(" anul: ");
scanf("%d",&data_nas.an);
printf("\n Specificati data angajarii: \n\n");
printf(" ziua: ");
scanf("%d",&data_ang.zi);
printf(" luna: ");
scanf("%s",data_ang.nume_luna);
printf(" anul: ");
scanf("%d",&data_ang.an);
printf("\n\n V-ati nascut la data:     %2d %-12s %4d \n\n si v-ati angajat la data:\
 %2d %-12s %4d\n",data_nas.zi,data_nas.nume_luna,data_nas.an,data_ang.zi,\
data_ang.nume_luna,data_ang.an);
getch();
```


Scanned with OKEN Scanner

```
}
```

B4) Aplicație pentru redenumirea tipului predefinit "pointer pe întreg": să se citească și să se afișeze cinci numere întregi. a căror adrese se află într-un tablou de pointeri pe întregi.

```
#include <conio.h>
#include <stdio.h>
typedef int *PI;

void main(void)
{
PI a[5];
int i;
clrscr();
printf("\n\n Introduceti 5 numere intregi\n");
for(i = 0; i < 5; ++i) {
        printf(" a[%d] = ",i);
        scanf("%d",a[i]);
        }
printf("\n Ati introdus numerele: \n\n");
for(i = 0; i < 5 ; i++)
        printf(" %5d ",*a[i]);
getch();
}
```

B5) Programul exemplifică modul de utilizare al structurilor. S-a dat ca exemplu o structură de tip personal care are ca membru o structură de tip data calendaristică. Să se inițializeze structura de tip personal și apoi să se afișeze elementele ei.

```
#include <conio.h>
#include <stdio.h>

struct data_calendar {
        int zi;
        char nume_luna[12];
        int an;
        } data_nas,data_ang;

struct personal {
```


Scanned with OKEN Scanner

```
        char nume_prenume[50];
        char adresa[50];
        struct data_calendar data_nas,data_ang;
        }pers = {" Popescu I. Ioan "," Aleea Energiei nr.17 Iasi", 19,"ianuarie"\
        ,1943,1,"septembrie",1966};
void main(void)
{
clrscr();
printf("\n\n Datele privind persoana: %-20s\n Adresa: %-20s\n data nasterii "
"%2d %-10s %4d si data angajarii %2d %-10s"
%4d\n",pers.nume_prenume,pers.adresa,pers.data_nas.zi,pers.data_nas.nume_luna,
pers.data_nas.an,pers.data_ang.zi,pers.data_ang.nume_luna,pers.data_ang.an);
printf("\n\n Initiala tatalui lui Popescu este  %c. \n",pers.nume_prenume[9]);
getch();
}
```

B6) Să se scrie un program prin care se citesc două şiruri de maxim 50 caractere şi să se determine poziţia în cadrul primului şir a primului caracter comun celor două şiruri.

Se vor folosi funcţii proprii atît pentru citirea şirului,cît şi pentru determinarea poziţiei cerută.

```
#include <stdio.h>
#include <conio.h>

#define DIMSIR 50
int pozitie(char s1[],char s2[]);
int getlinie(char s[],int n);

void main(void)
{
int poz;
char r,s1[DIMSIR],s2[DIMSIR];
do {
fflush(stdin);
clrscr();
printf("\n PROGRAMUL DETERMINA POZITIA IN CADRUL SIRULUI S1 A"
"PRIMULUI CARACTER COMUN \n                    INTRE SIRURILE S1 SI"
"S2\n");
printf(" ATENTIE: dimensiunea sirurilor este de maxim 50 de caractere!!\n");
```


Scanned with OKEN Scanner

```
printf("\n Introduceti primul sir:\n\n ");
getlinie(s1,DIMSIR);
printf("\n Introduceti al doilea sir:\n\n ");
getlinie(s2,DIMSIR);
printf("\n S-au introdus sirurile:\n s1: . %s\n s2:  %s\n",s1,s2);
poz = pozitie(s1,s2);
if(poz == -1)
        printf("\n Sirurile s1 si s2 nu au nici un caracter comun! \n");
else
        printf("\n Pozitia cautata in sirul s1:  %d  \n",poz + 1);
printf("\n Tastati D/d daca continuati:  ");
scanf(" %c",&r);
        } while(r == 'D' || r == 'd') ;
}

int pozitie(char s1[],char s2[])
{
int i,j;
i=0;
while(s2[i] != '\0')  {
        j = 0;
        while(s1[j] != '\0')  {
                if(s2[i] == s1[j])
                        return(j);
                else
                        j++;
                }
        i++;
}
return(-1);
}

int getlinie(char s[],int n)
{
int c,i=0;
while(--n > 0 && (c=getchar()) != '\n' && c != EOF)
        s[i++] = c;
if(c == '\n')
        s[i++] = c;
s[i] = '\0';
return i;
```


Scanned with OKEN Scanner

```
}
```

B7) Să se scrie şi să se testeze o funcţie recursivă pentru calculul factorialului unui număr.

```
#include <stdio.h>
#include <conio.h>

void fact(int n,long *);

void main()
{
int n;
long f;
char sf;
do {
clrscr();
printf("\n Specificati un nr. < 15 pt.care calculati factorialul:  ");
scanf("%d",&n);
fact(n,&f);
printf("\n\n %d!  =  %ld \n",n,f);
printf("\n Continuati? (D/d  sau  N/n)     ");
scanf(" %c",&sf);
} while(sf == 'd' || sf == 'D') ;
}

void fact(int n,long *rez)
{
if(n == 0)
        *rez = 1;
else  {
                        fact(n-1,rez);
                        *rez = n * (*rez);
                        }
}
```

B8) Să se citească elementele unei matrici pătratice de numere întregi. Să se afişeze matricea citită.Să se afişeze apoi suma elementelor de pe fiecare linie,coloană şi de pe cele două diagonale. Toate aceste operaţii vor fi realizate cu funcţii proprii.

Scanned with OKEN Scanner

```
#include <stdio.h>
#include <conio.h>

#define DIM 20

void cit_mat(int);
void af_mat(int);
int sum_col(int,int);
int sum_lin(int,int);
int sum_dp(int);
int sum_ds(int);

int mat[DIM][DIM];

void main(void)
{
int n,i,j;
clrscr();
printf("Precizati dimensiunea matricei patratice < 20  ");
scanf("%d",&n);
printf("Introduceti elementele matricii\n");
cit_mat(n);
printf("\nPentru a continua apasati orice tasta...");
getch();
clrscr();
printf("Ati introdus matricea:\n\n");
af_mat(n);
printf("\nPentru a continua apasati orice tasta...");
getch();
printf("\n\n");
for(j=0; j<n ; j++)
        printf("Suma pe coloana %2d este %3d\n",j,sum_col(j,n));
printf("\n");
for(i=0; i<n ; i++)
        printf("Suma pe linia %2d este %3d\n",i,sum_lin(i,n));
printf("\n");
printf("Suma pe diagonala principala este %3d\n\n",sum_dp(n));
printf("Suma pe diagonala secundara este %3d\n\n",sum_ds(n));
printf("Pentru a termina apasati orice tasta...");
getch();
}
```


Scanned with OKEN Scanner

```
void cit_mat(int n)
{
int i,j;
for(i=0; i<n ; i++)
        for(j=0; j<n ; j++)  {
                printf("mat[%2d][%2d] = ",i,j);
                scanf("%d",&mat[i][j]);
                }
}

void af_mat(int n)
{
int i,j;
for(i=0; i<n; i++) {
        for(j=0; j<n ; j++)
                printf(" %3d",mat[i][j]);
        printf("\n");
        }
}

int sum_col(int j,int n)
{
int i,s=0;
for(i=0; i<n ; i++)
        s += mat[i][j];
return s;
}

int sum_lin(int i,int n)
{
int j,s=0;
for(j=0; j<n ; j++)
        s += mat[i][j];
return s;
}

int sum_dp(int n)
{
int i,s=0;
for(i=0; i<n; i++)
```


Scanned with OKEN Scanner

```
        s += mat[i][i];
return s;
}

int sum_ds(int n)
{
int i,j,s=0;
for(i=n-1, j=0 ; i>=0 && j<n ; --i,++j)
        s += mat[i][j];
return s;
}
```

B9) Să se scrie un program pentru calculul modulelor unui şir de numere complexe.Ieşirea din bucla de citire a numerelor complexe se face tastînd orice caracter nenumeric.

Numărul complex se defineşte ca o structură,iar funcţiei care calculează modulul i se transmite un pointer pe această structură.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

typedef struct  {
        double real,imaginar;
        } COMPLEX;

double modul(COMPLEX *);

void main(void)
{
COMPLEX a;
clrscr();
printf(" \n Pentru iesire din bucla de citire date,tastati orice caracter"
      " nenumeric...\n");
printf(" \n Introduceti cite un numar complex astfel: partea reala  partea"
      " imaginara\n");
while(scanf("%lf %lf",&a.real,&a.imaginar) == 2)
        printf("\n Modulul numarului %.2lf + %.2lf i  =  %.2lf"
              " \n",a.real,a.imaginar,modul(&a));
}
```


Scanned with OKEN Scanner

```
double modul(COMPLEX *x)
{
return(sqrt(x->real * x->real + x->imaginar * x->imaginar));
}
```

B10) Să se scrie un program pentru calculul modulului unui număr complex, definit ca o structură. Funcţia "modul" care face calculul cerut,primeşte ca parametri cei doi membri ai structurii definite.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

typedef struct  {
        double real,imaginar;
        } COMPLEX;

double modul(double,double);

void main(void)
{
COMPLEX a;
double mod;
char sf;
do  {
clrscr();
printf(" \n Introduceti un numar complex:\n");
printf("\n Partea reala:  ");
scanf("%lf",&a.real);
printf("\n Partea imaginara:  ");
scanf("%lf",&a.imaginar);
mod = modul(a.real,a.imaginar);
printf("\n Modulul numarului %.2lf + %.2lf i  =  %.2lf \n",a.real,a.imaginar,mod);
printf("\n Continuati d/n   ?   ");
while(( sf = getchar()) != 'd' && sf != 'D' && sf != 'n' && sf != 'N') ;
} while(sf == 'd' || sf == 'D') ;
}

double modul(double r,double i)
```


Scanned with OKEN Scanner

```
{
return(sqrt(r * r + i * i));
}
```

B11) Să se scrie un program prin care se citeşte linie cu linie un text din intrarea standard. In fiecare linie se caută un model, dat ca argument în linia de comandă. Dacă în linia de comandă se dă şi argumentul optional -x se tipăresc doar liniile care nu conţin modelul. Dacă în linia de comandă se dă argumentul opţional -xn sau -nx se tipăresc liniile care nu conţin modelul, cu numărul de ordine al liniei. Dacă în linia de comandă se dă argumentul opţional -n se tipăresc doar liniile care conţin modelul avînd în faţă numărul de ordine al liniei. Dacă nu există nici un argument opţional, se tipăresc liniile care conţin modelul.

Exemplu de linie de comandă: -xn abba

```
#include <stdio.h>
#include <process.h>
#include <conio.h>

#define LMAX 100

int getlinie(char s[],int n);
int index(char s[],char *t);

void main(int argc,char *argv[])
{
char *s;
char linie[LMAX];  /* linia citita */
int nl=0;          /* numarul liniei */
int except=0;      /* daca este 1 se tiparesc liniile ce nu contin modelul*/
int nr=0;          /* daca este 1 se tipareste numarul liniei */
clrscr();
while(--argc > 0 && (*++argv)[0] == '-')  /* test ptr.arg.optional */
        for(s=argv[0]+1; *s != '\0' ;s++ )
                switch(*s) {
                        case 'x':  except = 1;  break;
                        case 'n':  nr=1;  break;
                        default:
                         printf("\n Optiune ilegala in linia de comanda %c\n",*s);
                         argc=0; break;
                        }
```


Scanned with OKEN Scanner

```
if(argc != 1) {
        printf("\n Utilizare eronata a modelului!\n");
        getch();
        exit(1);
        }
else {
        printf("\n Introduceti liniile cu si fara modelul: %s\n",argv[0]);
        while(getlinie(linie,LMAX) > 0 ) {
                nl++;
                if(except == 1) {
                        printf("\n Se tiparesc liniile care nu contin modelul!\n\n");
                        if(index(linie,argv[0]) < 0) {
                                if(nr)
                                        printf("%2d: ",nl);
                                printf("%s",linie);
                                }
                        }
                else {
                        printf("\n Se tiparesc liniile care contin modelul!\n\n");
                        if(index(linie,argv[0]) >= 0) {
                                if(nr)
                                        printf("%2d: ",nl);
                                printf("%s",linie);
                                }
                         }
                }
        }
}

int getlinie(char s[],int n)
{
int c,i=0;
while(--n > 0 && (c=getchar()) != '\n' && c != EOF)
        s[i++] = c;
if(c == '\n')
        s[i++] = c;
s[i] = '\0';
return i;
}

int index(char s[], char *t)
```


Scanned with OKEN Scanner

```
{
int i,j;
for(i=0; s[i] != '\0'; i++)  {
        for(j=i; *t != '\0' && s[j] == *t; j++,t++) ;
        if(*t == '\0')
                return i;
        }
return -1;
}
```

B12) Să se scrie un program care realizează următoarele:

- citeşte cîte o linie din intrarea standard, folosind o funcţie proprie;
- caută în fiecare linie citită un model (subşir), dat ca argument în linia de comandă. Linia care conţine modelul va fi tiparită. Pentru căutarea modelului va fi scrisă o funcţie proprie.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

#define LMAX 100

int getlinie(char s[],int n);
int index(char s[],char *t);

void main(int argc,char *argv[])
{
char linie[LMAX];
clrscr();
printf("\n Introduceti liniile cu si fara model:\n\n");
if(argc != 2)  {
        printf("\n Modelul nu este dat corect sau nu este dat deloc!\n");
        getch();
        exit(1);
        }
else
        while(getlinie(linie,LMAX) > 0)
                if(index(linie,argv[1]) >= 0)
                        printf("%s",linie);
}
```


Scanned with OKEN Scanner

```
int getlinie(char s[],int n)
{
int c,i=0;
while(--n > 0 && (c=getchar()) != '\n' && c != EOF)
        s[i++] = c;
if(c == '\n')
        s[i++] = c;
s[i] = '\0';
return i;
}

int index(char s[], char *t)
{
int i,j;
for(i=0; s[i] != '\0'; i++)  {
        for(j=i; *t != '\0' && s[j] == *t; j++,t++) ;
        if(*t == '\0')
                return i;
        }
return -1;
}
```

B13) Să se scrie un program care calculează ariile unor figuri geometrice. Mai întîi se specifică pentru ce figură se calculează aria. Apoi se citesc elementele figurii, cu validările necesare. Dacă datele sunt corect citite, se calculează aria şi se afişează. Programul se poate relua la cerere. Elementele care definesc o figură constituie o uniune.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>

#define CERC            1
#define PATRAT          2
#define DREPTUNGHI      3
#define TRIUNGHI        4
#define EROARE          -1
#define PI              3.1416
```


Scanned with OKEN Scanner

```
typedef struct figura {
        int tip_fig;
        union  {
                double raza;            /* cerc      */
                double lat_pat;         /* patrat    */
                double lat_drept[2];    /* dreprunghi */
                double triunghi[3];     /* triunghi  */
                } fig;
        } FIG;

double arie(FIG *);
int cit_fig(FIG *);

void main(void)
{
char r,sf;
double supr;
FIG f;
do {
clrscr();
printf(" ATENTIE: Elementele figurilor sunt NUMERICE!\n\n");
printf(" Tastati C/c daca doriti aria unui CERC\n Tastati P/p daca doriti aria unui"
       " PATRAT \n Tastati D/d daca doriti aria unui DREPTUNGHI\n Tastati T/t"
       " daca doriti aria unui triunghi:\n        ");
scanf(" %c",&r);
for( ; ; ) {
        switch(r) {
                case 'C':
                case 'c':
                        f.tip_fig = CERC;
                        break;
                case 'P':
                case 'p':
                        f.tip_fig = PATRAT;
                        break;
                case 'D':
                case 'd':
                        f.tip_fig = DREPTUNGHI;
                        break;
                case 'T':
```


Scanned with OKEN Scanner

```
                case 't':
                        f.tip_fig = TRIUNGHI;
                        break;
                default:
                        printf("\n Nu s-a tascat corect tipul figurii geometrice!\n");
                        f.tip_fig = EROARE;
                }
                break;
        } /* sfirsit de for */
if(cit_fig(&f) != 0) {
        supr = arie(&f);
        printf(" Aria figurii:  %.3lf",supr);
        }
printf("\n\n Doriti sa continuati programul?  Y/y  ");
scanf(" %c",&sf);
}while(sf == 'Y' || sf == 'y');
}

double arie(FIG *p)
{
double sp,a,b,c;
switch(p->tip_fig)  {
        case CERC:
                return PI * p->fig.raza * p->fig.raza;
        case PATRAT:
                return p->fig.lat_pat * p->fig.lat_pat;
        case DREPTUNGHI:
                return p->fig.lat_drept[0] *p->fig.lat_drept[1];
        case TRIUNGHI:
                a = p->fig.triunghi[0];
                b = p->fig.triunghi[1];
                c = p->fig.triunghi[2];
                sp = (a+b+c)/2;
                return sqrt(sp*(sp-a)*(sp-b)*(sp-c));
        default:
                return 0;          /* eroare */
        }
}

int cit_fig(FIG *p)
{
```


Scanned with OKEN Scanner

```
int exp;
double a,b,c;
switch(p->tip_fig) {
        case CERC:
                do {
                        printf("\n Introduceti raza cercului:  ");
                        scanf("%lf",&a);
                        if(a > 0.)
                                p->fig.raza = a;
                        else
                                printf("\n Raza este gresit citita!\n");
                }while(a <= 0.);
                return 1;
        case PATRAT:
                do {
                        printf("\n Introduceti latura patratului:  ");
                        scanf("%lf",&a);
                        if(a > 0.)
                                p->fig.lat_pat = a;
                        else
                                printf("\n Latura patratului este citita gresit!\n");
                }while(a <= 0);
                return 1;
        case DREPTUNGHI:
                do {
                        printf("\n Introduceti cele doua laturi ale dreptunghiului:   ");
                        scanf("%lf %lf",&a,&b);
                        if(a > 0 && b > 0) {
                                p->fig.lat_drept[0] = a;
                                p->fig.lat_drept[1] = b;
                                }
                        else
                                printf("\n Laturile dreptunghiului sunt citite gresit!\n");
                }while(a <= 0. || b <= 0.);
                return 1;
        case TRIUNGHI:
                do {
                        printf("\n Introduceti cele trei laturi ale triunghiului:   ");
                        scanf("%lf %lf %lf",&a,&b,&c);
                        exp = (a < b+c && b < a+c && c < b+a);
                        if(exp) {
```


Scanned with OKEN Scanner

```
                                p->fig.triunghi[0] = a;
                                p->fig.triunghi[1] = b;
                                p->fig.triunghi[2] = c;
                                }
                        else
                                printf("\n Date eronate pentru triunghi!\n");
                }while(exp == 0);
                return 1;
        default:
                printf("\n Figura necunoscuta de program!\n");
                return 0;
        }
}
```

B14) Să se scrie şi să se testeze doua funcţii pentru conversia datei calendaristice:
- se dă ziua, luna şi anul şi să se calculeze a cîta zi din an este;
- se dă ziua din an şi anul şi să se calculeze luna şi ziua din lună.

Pentru data calendaristică se va defini o structură.

```
#include <stdio.h>
#include <conio.h>

struct data {
                int zi,luna,an,zian;
                } d;

int zi_din_an(struct data *);
void zi_din_luna(struct data *);

static int tab_zile[2][13] = {
                {0,31,28,31,30,31,30,31,31,30,31,30,31},
                {0,31,29,31,30,31,30,31,31,30,31,30,31}
                };

void main(void)
{
char c;
do {
        clrscr();
        printf("\n Introduceti ziua,luna si anul pentru care doriti ziua din an:\n\n");
```


Scanned with OKEN Scanner

```
        printf(" ziua = ");
        scanf("%d",&d.zi);
        printf(" luna = ");
        scanf("%d",&d.luna);
        printf(" anul = ");
        scanf("%d",&d.an);
        d.zian = zi_din_an(&d);
        printf("\n Ziua din an este: %3d  iar luna  %2d ",d.zian,d.luna);
        printf("\n\n Introduceti ziua din an si anul pentru care doriti ziua si"
               " luna:\n\n");
        printf(" ziua din an = ");
        scanf("%d",&d.zian);
        printf(" anul = ");
        scanf("%d",&d.an);
        zi_din_luna(&d);
        printf("\n ziua = %2d  luna = %2d  anul %4d\n",d.zi,d.luna,d.an);
        printf("\n Doriti sa reluati?  D/d  ");
        scanf(" %c",&c);
        }while(c == 'D' || c == 'd');
}

int zi_din_an(struct data *pd)
{
int i,zi,bis;
zi = pd->zi;
bis = pd->an % 4 == 0 && pd->an %100 != 0 || pd->an % 400 == 0;
for(i=1; i < pd->luna; i++)
        zi += tab_zile[bis][i];
return zi;
}

void zi_din_luna(struct data *pd)
{
int i,bis;
bis = pd->an % 4 == 0 && pd->an %100 != 0 || pd->an % 400 == 0;
pd->zi = pd->zian;
for(i=1; pd->zi > tab_zile[bis][i]; i++)
        pd->zi -= tab_zile[bis][i];
pd->luna = i;
}
```


Scanned with OKEN Scanner

B15) Să se scrie un program în care se lucrează cu o matrice declarată static ca având 5 x 5 elemente numere întregi.Se va declara şi un tablou de pointeri pe 5 şiruri de întregi, din care primele trei şiruri coincid cu primele trei linii ale matricii.

Să se citească elementele acestor şiruri şi să se afişeze utilizînd tabelul de pointeri şi apoi utilizînd matricea.

Să se compare declararea utilizînd tabloul multidimensional şi tabloul de pointeri.

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>

int mat[5][5];
int *tab[5] = { mat[0],mat[1],mat[2]};     //,mat[3],mat[4]};

void main(void)
{
int i,j;
clrscr();
printf("\n Introdu numerele din primele trei linii:\n");
for(j=0; j<3; j++)
        for(i=0; i<5; i++) {
                printf(" *tab[%d] + %d = ",j,i);
                scanf("%d",tab[j]+i);
        }
printf("\n Introdu numerele din urmatoarele doua linii:\n");
for(j=3; j<5; j++) {
        tab[j] = (int *)malloc(5*sizeof(int));
        for(i=0; i<5; i++) {
                printf(" *tab[%d] + %d = ",j,i);
                scanf("%d",tab[j]+i);
                }
        }
printf("\n Ai introdus liniile - tiparirea se face utilizind tabelul de pointeri:\n\n");
for(j=0; j<5; j++)  {
        for(i=0; i<5;i++)
                printf("%4d",*(tab[j]+i));
        printf("\n");
        }
printf("\n Ai introdus liniile - tiparirea se face utilizind matricea:\n\n");
for(j=0; j<5; j++) {
```


Scanned with OKEN Scanner

```
        for(i=0; i<5; i++)
                printf("%4d",mat[j][i]);
        printf("\n");
        }
getch();
}
```

B16) Să se scrie un program prin care se citeşte un număr întreg din intrarea standard şi se returnează numele lunii corespunzătoare acelui număr sau 'luna eronata' dacă este cazul.

Se va folosi un tablou de pointeri pe numele lunilor.

```
#include <stdio.h>
#include <conio.h>

char *nume_luna(int);

void main(void)
{
int nr;
char r;
do {
        clrscr();
        printf("\n Dati numarul corespunzator lunii:  ");
        scanf("%d",&nr);
        printf("\n Luna se numeste:  %s\n",nume_luna(nr));
        printf("\n Continuati?  Atunci tastati D/d... ");
        scanf(" %c",&r);
} while(r == 'D' || r == 'd');
}

char *nume_luna(int n)
{
static char *nume[] = {
        "Luna eronata",
        "Ianuarie",
        "Februarie",
        "Martie",
        "Aprilie",
        "Mai",
```


Scanned with OKEN Scanner

```
        "Iunie",
        "Iulie",
        "August",
        "Septembrie",
        "Octombrie",
        "Noiembrie",
        "Decembrie"
        };
return(( n < 1 || n > 12 ) ? nume[0] : nume[n]);
}
```

B17) Să se scrie un program pentru creerea unui tabel de simboli care conţine numele unei constante simbolice şi valoarea ei. Tabelul de simboli va fi organizat ca un tabel de dispersie .

Numele constantei simbolice va respecta restricţiile numelui oricărui identificator : va începe cu o literă sau cu '_' şi va conţine caractere alfanumerice sau '_'. Orice şir care nu respectă aceste restricţii va fi ignorat.

Va fi testată o funcţie de căutare a numelui în tabel şi o functie de instalare a perechii (nume,valoare) în tabelul de dispersie.

Coliziunile, adică cheile cu acciaşi funcţie de dispersie vor fi salvate în liste simplu înlănţuite.

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#include <string.h>
#include <stdlib.h>

#define MAXS        30
#define MAXV        100
#define M           127
#define LITERA      'a'
#define NR          '0'

struct lista  {
        char *nume,*val;
        struct lista *urm;
        };

struct lista *tab_has[M];
```


Scanned with OKEN Scanner

```
int hash(char *);
int getcuv(char [],int);
int     tip(int);
void init(void);
void elib(void);
void afis(void);
struct lista *cauta(char *);
struct lista *instalare(char *,char *);

void main(void)
{
char sf,nume[MAXS],val[MAXV];
int t;
struct lista *pl;
init();
clrscr();
printf("\n Se creeaza o tabela de simboli continind perechile (nume,valoare)\n");
printf("\n ATENTIE: Iesirea din bucla de citire se face tastand GATA  \n");
printf("\n Dati primul nume din tabela:     ");
t = getcuv(nume,MAXS);
while((strcmp(nume,"GATA")) != 0 && t != EOF)
        if(t==LITERA) {
                printf(" Introduceti valoarea numelui:    ");
                gets(val);
                instalare(nume,val);
                printf(" Dati urmatorul nume din tabela:  ");
                t = getcuv(nume,MAXS);
                }
        else {
                printf(" Dati va rog numele corect:       ");
                t = getcuv(nume,MAXS);
                }
printf("\n Continutul tabelei de simboli:\n");
afis();
do {
        printf("\n Dati un nume de cautat:         ");
        if(getcuv(nume,MAXS)==LITERA)
                if((pl = cauta(nume)) != NULL) {
                        printf("\n Numele  %s  este in lista si are valoarea  %s ",
                                nume,pl->val);
```


Scanned with OKEN Scanner

```
                    getch();
                    }
                else{
                    printf("\n Numele nu este in lista dar se instaleaza cu valoarea: ");
                    gets(val);
                    instalare(nume,val);
                    afis();
                    }
        else
                printf(" \n Numele este incorect\n");
        printf("\n Daca mai doriti o cautare,tastati D/d  ");
        scanf(" %c",&sf);
        getchar();
} while(sf == 'D' || sf == 'd');
clib();
}

int hash(char *s)
{
int h;
for(h=0; *s != '\0';)
        h += *s++;
return h%M;
}

struct lista *cauta(char *s)
{
struct lista *pl;
for(pl=tab_has[hash(s)]; pl != NULL ; pl = pl->urm)
        if(strcmp(s,pl->nume) == 0)
                return pl;    /* este gasita cheia; intoarce pointer pe ea */
return NULL;        /* nu este gasita cheia */
}

struct lista *instalare(char *s,char *t)
{
struct lista *pl;
int h;
if(( pl = cauta(s)) == NULL)  { /* nu gaseste cheia */
        pl = (struct lista *)malloc(sizeof(struct lista));
        if(pl == NULL) {
```


Scanned with OKEN Scanner

```
                printf("\n Nu se poate aloca memorie pentru nod!\n");
                getch();
                exit(1);
                }
        h = hash(s);
        pl->nume = (char *)malloc(strlen(s)+1);
        pl->val = (char *)malloc(strlen(t)+1);
        strcpy(pl->nume,s);
        strcpy(pl->val,t);
        pl->urm = tab_has[h];
        tab_has[h] = pl;
        }
else {  /* cheia cautata este intr-o lista; se actualizeaza valoarea*/
        free(pl->val);
        pl->val=(char *)malloc(strlen(t)+1);
        strcpy(pl->val,t);
        }
return pl;
}

void afis(void)
{
struct lista *pl;
int i;
for(i=0; i < M; i++)
        for(pl = tab_has[i]; pl != NULL; pl = pl->urm)
                if(tab_has[i] != NULL)
                        printf("pt.h = %3d simbol: %-20s  valoare: %-30s",
                                i,pl->nume,pl->val);
}

void elib(void)
{
struct lista *pl;
int i;
for(i=0; i < M; i++)
        for(pl=tab_has[i]; pl != NULL; pl = pl->urm) {
                free(pl->nume);
                free(pl->val);
                free(pl);
                }
```


Scanned with OKEN Scanner

```
}

void init(void)
{
int i;
for(i=0; i < M; i++)
        tab_has[i] = NULL;
}

int getcuv(char cuv[],int lim)
{
int c,t,i=0;
c = getchar();
if(c == EOF)
        return c;
for(i=0; i<lim-1 && c != ' ' && c != '\n' && c != '\t' ; ++i) {
        cuv[i] = c;
        c = getchar();
        }
cuv[i] = '\0';
i=0;
if(( t = tip(cuv[0])) != LITERA ) {
        cuv[1] = '\0';
        return cuv[0];
        }
else  {
        i = 1;
        while((t = tip(cuv[i++])) != '\0') {
                if(t == LITERA || t == NR)
                        continue;
                else
                        return cuv[--i];
                }
        }
return LITERA;
}

int tip(int c)
{
if(c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z' || c == '_')
        return(LITERA);
```


Scanned with OKEN Scanner

```
else
        if(c >= '0' && c <= '9')
                return(NR);
        else
                return(c);
}
```

B18) Stivă realizată printr-o lista simplu înlănţuită:datele se introduc şi se scot prin vîrful stivei - principiul LIFO. Vîrful stivei este nodul spre care pointează variabila "prim", iar baza nodul spre care pointează variabila "ultim".

Aplicaţie: Se consideră într-o gară un tren de marfă ale cărui vagoane sînt inventariate într-o listă, care conţine, în ordinea vagoanelor, date specifice acestora : cod vagon şi cod marfă cu maxim 10 caractere, adresa expeditorului şi adresa destinaţiei cu maxim 40 caractere. Deoarece în gară se inversează poziţia vagoanelor, se cere listarea datelor despre vagoane în ordine inversă.

```
#include <stdio.h>
#include <alloc.h>
#include <conio.h>
#include <string.h>

struct tnod  {
        char codvag[11],codmarfa[11],adrexp[41],adrdest[41];
        struct tnod *urm;
        };

struct tnod *prim,*ultim;

struct tnod *afis(void);
struct tnod *insert_prim(int);
void incarca_nod(struct tnod *,int);
void sterg_virf(void);
void cit_adresa(char []);

void main(void)
{
int i = 0,n;
prim = ultim = NULL; /* lista vida */
/* se creeaza stiva de tip lista simplu inlantuita inserind cite un nod in fata primului
cu functia insert_prim */
```


Scanned with OKEN Scanner

```
clrscr();
printf("\n Specificati pentru cite vagoane introduceti date:  ");
scanf("%d",&n);
for(i=0; i<n; i++)
        if(insert_prim(i) != NULL)
                continue;
        else {
                printf("\n memorie insuficienta pentru inserarea unui nod\n");
                getch();
                }
/* se listeaza cite un element de pe stiva si apoi se iese de pe stiva */
i=0;
while(prim != NULL)  {
        i++;
        printf("\n Vagonul %2d\n",i);
        printf(" - cod vagon:  %-10s  - cod marfa:   %-10s\n",prim->codvag,
              prim->codmarfa);
        printf(" - adresa expeditorului:  %-40s\n",prim->adrexp);
        printf(" - adresa destinatarului: %-40s\n",prim->adrdest);
        sterg_virf();  /* sterge nodul din virful stivei */
        }
getch();
}

/* functie de inserare inaintea nodului spre care pointeaza prim */
struct tnod *insert_prim(int nv)
{
struct tnod *nod_nou;
int n;
n = sizeof(struct tnod);
if((nod_nou = (struct tnod *)malloc(n)) != NULL)   {
                incarca_nod(nod_nou,nv);
                if(prim == NULL)  {
                        prim = ultim = nod_nou;
                        prim->urm = NULL;
                        }
                else {
                        nod_nou->urm = prim;
                        prim = nod_nou;
                        }
                return nod_nou;
```


Scanned with OKEN Scanner

```
        }
else
                return NULL;
}

/* functie de incarcare a nodului curent cu datele vagonului */
void incarca_nod(struct tnod *nod_nou,int nv)
{
printf(" Introduceti datele pentru vagonul %d:\n",nv+1);
printf(" - cod vagon (max.10 caractere):   ");
scanf("%s",nod_nou->codvag);
printf(" - cod marfa (max.10 caractere):   ");
scanf("%s",nod_nou->codmarfa);
getchar();
printf(" - adresa exp.(max.40 caractere):  ");
cit_adresa(nod_nou->adrexp);
printf(" - adresa dest.(max.40 caractere): ");
cit_adresa(nod_nou->adrdest);
}

/* functie pentru citirea adresei */
void cit_adresa(char s[])
{
int c,i;
for(i=0; i<40 && (c=getchar()) != '\n'; i++)
        s[i]=c;
s[i] = '\0';
}

/* functie de stergere a primului nod din lista */
void sterg_virf(void)
{
struct tnod *p;
if(prim == NULL)  {
        printf("lista vida \n");
        return;
        }
p = prim;
prim = prim->urm;
free(p);
if(prim == NULL)
```


Scanned with OKEN Scanner

```
        ultim = NULL;
}
```

B19) Să se creeze o listă dublu înlănțuită, care are ca informații utile numere întregi, diferite de zero. Lista va fi creată prin inserție, astfel ca informațiile din noduri să fie în ordine crescătoare și să nu se repete. Să se testeze și o funcție de inversare a nodurilor listei. Lista va fi afișată înainte și după inversare.

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>

struct list  {
        int inf;
        struct list *prec;
        struct list *urm;
        }  *prim;

void insert(int);
void afis(void);
void invers(void);

void main(void)
{
int x;
clrscr();
printf("\n Creerea unei liste dublu inlantuite avind ca elemente numere intregi\n");
printf(" ATENTIE! Iesirea din operatia de creere se face tastind 0 (zero)\n\n");
printf(" Introduceti primul element al listei:    ");
scanf("%d",&x);
while(x != 0)  {
        insert(x);
        printf(" introduceti urmatorul element din lista: ");
        scanf("%d",&x);
        }
printf("\n Elementele listei sunt:\n\n");
afis();
invers();
printf("\n\n Lista inversata:\n\n");
afis();
```


Scanned with OKEN Scanner

```
getch();
}

void insert(int u)
{
struct list *nod_nou,*urmator,*precedent;
if(( nod_nou = (struct list*)malloc(sizeof(struct list))) == NULL) {
        printf("\n Nu poate fi alocat spatiu!\n");
        return;
        }
nod_nou->inf = u;
nod_nou->prec = nod_nou->urm = NULL;
if(prim == NULL)  {
        prim = nod_nou;
        return;
        }
for(urmator = prim; urmator != NULL && urmator->inf < u;
                        urmator = urmator->urm)
        precedent = urmator;
if(urmator->inf == u)  {  /* informatia din nod se repeta */
        free(nod_nou);
        return;
        }
if(urmator == prim)  { /*nodul nou se insereaza in fata primului*/
        urmator->prec = nod_nou;
        nod_nou->urm = prim;
        prim = nod_nou;
        prim->prec = NULL;
        }
else  {      /* nodul nou se insereaza intre doua noduri */
        nod_nou->urm = precedent->urm;
        precedent->urm = nod_nou;
        urmator->prec = nod_nou;
        nod_nou->prec = precedent;
        }
return;
}

void afis(void)
{
struct list *adr;
```


Scanned with OKEN Scanner

```
for(adr = prim; adr != NULL; adr = adr->urm)
        printf("  %d <--> ",adr->inf);
}

void invers(void)
{
int t,n = 1;
struct list *a,*b,*c;
for(a = prim; a != NULL; n++, a = a->urm)
        c = a;
n = n/2;
for(a = prim, b = c; n-- >0 && a->inf < b->inf; a = a->urm,b = b->prec)  {
        t = a->inf;
        a->inf = b->inf;
        b->inf = t;
        }
}
```

B20) Program pentru căutarea unor cuvinte cheie într-un text. Se consideră cuvînt o secvență de caractere alfanumerice care începe cu o literă sau cu '_', conține litere, caractere numerice sau '_' şi care este cuprinsă între caractere de spațiere. Funcția de căutare întoarce un pointer pe o structură de tip "chei".

```
#include <stdio.h>
#include <conio.h>
#include <string.h>

#define LITERA          'a'
#define NR              0
#define MAXCUV          20
#define NKEY            (sizeof(tab_chei)/sizeof(struct chei))

struct chei {
        char *cuv_ch;
        int contor;
        } tab_chei[] = {
                "break",0,
                "case",0,
                "char",0,
                "continue",0,
```


Scanned with OKEN Scanner

```
                "default",0,
                "do",0,
                "double",0,
                "else",0,
                "float",0,
                "for",0,
                "if",0,
                "int",0,
                "long",0,
                "main",0,
                "register",0,
                "short",0,
                "static",0,
                "struct",0,
                "switch",0,
                "union",0,
                "unsigned",0,
                "while",0,
                };

int getcuv(char [],int );
int tip(int );
struct chei *binar(char *,struct chei [],int);

void main(void)
{
int t;
char cuv[MAXCUV];
struct chei *p;
clrscr();
printf("\n Introduceti textul:\n\n");
while((t = getcuv(cuv,MAXCUV)) != EOF)
        if(t == LITERA)
                if((p = binar(cuv,tab_chei,NKEY)) != NULL)
                        p->contor++;
printf("\n\n Ati introdus urmatoarele cuvinte cheie:\n\n");
for(p = tab_chei; p < tab_chei + NKEY; p++)
        if( p->contor > 0)
                printf("cuvintul %-10s apare de %3d ori\n",p->cuv_ch,p->contor);
getch();
}
```


Scanned with OKEN Scanner

```
int getcuv(char cuv[],int lim)
{
int c,t,i;
c = getchar();
for(i=0; i<lim && c != ' ' && c != '\n' && c != '\t' ; ++i) {
        cuv[i] = c;
        c = getchar();
        }
cuv[i] = '\0';
i=0;
if(( t = tip(cuv[0])) != LITERA) {
        cuv[1] = '\0';
        return cuv[0];
        }
else  {
        i = 1;
        while((t = tip(cuv[i++])) != '\0') {
                if(t == LITERA || t == NR)
                        continue;
                else
                        return cuv[--i];
                }
        }
return LITERA;
}

int tip(int c)
{
if(c >= 'a' && c <= 'z'  || c >= 'A' && c <= 'Z' || c == '_')
        return(LITERA);
else
        if(c >= '0' && c <= '9')
                return(NR);
        else
                return(c);
}

struct chei *binar(char *cuv,struct chei tab[],int n)
{
int cmp;
```


Scanned with OKEN Scanner

```
struct chei *inf = &tab[0],*sup = &tab[n-1],*med;
while(inf <= sup)  {
        med = inf + (sup-inf)/2;
        if((cmp = strcmp(cuv,med->cuv_ch)) < 0)
                sup = med - 1;
        else
                if(cmp > 0)
                        inf = med + 1;
                else
                        return med;
        }
return NULL;
}
```

B21) Să se scrie un program pentru contorizarea numerelor de aparitii ale cuvintelor unui text citit de la tastatură. Textul va fi terminat cu un cuvînt terminal citit tot de la tastatură.

Se consideră cuvînt o secvenţă de caractere alfanumerice care începe cu o literă sau cu '_', conţine litere,caractere numerice sau '_' şi care este cuprinsă între caractere de spaţiere. Sirurile de caractere care nu respectă aceste restricţii vor fi ignorate.

Pentru rezolvarea problemei se va construi un arbore binar de căutare ale cărui noduri au ca informaţie utilă un pointer pe cuvînt şi numărul de apariţii ale cuvîntului.

```
#include <conio.h>
#include <stdio.h>
#include <alloc.h>
#include <string.h>
#include <stdlib.h>

#define MAXCUV 21
#define LITERA 'a'
#define NR '0'

struct nodarb  {
        char *cuvint;      /* pointeaza un cuvint din text */
        int contor;       /* contorizeaza cuvintul */
        struct nodarb *st;
        struct nodarb *dr;
```


Scanned with OKEN Scanner

```
        } ;

struct nodarb *arb(struct nodarb *,char *);
void arbprint(struct nodarb *);
void elib(struct nodarb *);
int      getcuv(char *,int );
int      tip(int);

void main(void)
{
struct nodarb *rad;
char cuv[MAXCUV],term[MAXCUV];
int t;
clrscr();
printf(" Introduceti cuvintul terminal al textului: ");
getcuv(term,MAXCUV);
printf("\n Introduceti cuvintele textului:\n\n");
rad = NULL;
t = getcuv(cuv,MAXCUV);
while(strcmp(term,cuv) != 0 && t != EOF){
        if(t == LITERA)
                rad = arb(rad,cuv);
        t = getcuv(cuv,MAXCUV);
        }
printf("\n Frecventa de aparitie a cuvintelor:\n");
arbprint(rad);
elib(rad);
getch();
}

struct nodarb *arb(struct nodarb *p,char *cuv)
{
int cond;
if(p == NULL)  {
        if((p = (struct nodarb *)malloc(sizeof(struct nodarb))) == NULL) {
                printf("\n Nu se poate aloca memorie pentru nod\n");
                getch();
                exit(1);
                }
        else {
                p->cuvint = (char *)malloc(strlen(cuv)+1);
```


Scanned with OKEN Scanner

```
                strcpy(p->cuvint,cuv);
                p->contor = 1;
                p->st = p->dr = NULL;
                }
        }
else {
        if(( cond = strcmp(cuv,p->cuvint)) == 0)
                p->contor++;
        else
                if( cond < 0 )
                        p->st = arb(p->st,cuv);
                else
                        p->dr = arb(p->dr,cuv);
        }
return(p);
}

void arbprint(struct nodarb *p)
{
if(p != NULL)  {
        arbprint(p->st);
        printf("\n cuvintul  %-20s  apare de %4d  ori",p->cuvint,p->contor);
        arbprint(p->dr);
        }
}

void elib(struct nodarb *p)
{
if(p != NULL) {
        elib(p->st);
        free(p->cuvint);
        free(p);
        elib(p->dr);
        }
}

int getcuv(char cuv[],int lim)
{
int c,t,i=0;
c = getchar();
if(c == EOF)
```


Scanned with OKEN Scanner

```
        return c;
for(i=0; i<lim-1 && c != ' ' && c != '\n' && c != '\t' ; ++i) {
        cuv[i] = c;
        c = getchar();
        }
cuv[i] = '\0';
i=0;
if(( t = tip(cuv[0])) != LITERA) {
        cuv[1] = '\0';
        return cuv[0];
        }
else {
        i = 1;
        while((t = tip(cuv[i++])) != '\0') {
                if(t == LITERA || t == NR)
                        continue;
                else
                        return cuv[--i];
                }
        }
return LITERA;
}
```

/* este o alta varianta de functie pentru citirea unui cuvint,dar mai putin restrictionata: un cuvint este un sir alfanumeric care incepe cu o litera,contine litere si cifre si tine dela prima litera a sirului pana la primul caracter diferit de un caracter alfanumeric.De ex. sirul asd%qwe reprezinta doua cuvinte: asd si qwe; la fel sirul las este cuvintul as, ceeace nu se intimpla la prima varianta */

```
/*int     getcuv(char *cuv,int lim)
{
int c,t;
if(tip(c = *cuv++ = getchar()) != LITERA)  {
        *cuv = '\0';
        return(c);
        }
while(--lim > 0)  {
        t = tip(c = *cuv++ = getchar()) ;
        if(t != LITERA && t != NR )
                break;
        }
```


Scanned with OKEN Scanner

```
*(cuv - 1) = '\0';
return(LITERA);
} */

int     tip(int c)
{
if(c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z')
        return(LITERA);
else
        if(c >= '0' && c <= '9')
                return(NR);
        else
                return(c);
}
```

B22) Să se citească de la intrarea standard datele ale căror formate sînt prezentate mai jos. Să se scrie aceste date în fişierul misc.dat din directorul curent. Să se afişeze conţinutul acestui fişier. Se va face prelucrarea unui fişier binar. Formatul datelor de intrare este următorul :

```
tip_doc  denumire        um   cod          pret   cantitate
   I     REZISTENTA      ko   123456789    1.5     10000.0
```

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>

#define MAX 30

typedef struct  {
        char tip_doc[5];
        char den[MAX + 1];
        char um[4];
        long cod;
        float pret;
        float cant;
        } ARTICOL;

union  {
```


Scanned with OKEN Scanner

```
        ARTICOL a[6];    /* 6 articole in zona tampon */
        char zt[6 * sizeof(ARTICOL) ];
} buf;

void afis(FILE *);
int cit(ARTICOL *);

void main(void) /* creaza fisierul misc.dat cu datele citite din stdin */
{
FILE *pf;
ARTICOL a;
int i,n;
if((pf = fopen("misc.dat","wb")) == NULL)  {   /*se deschide fis.in scriere binara*/
        printf(" Nu se poate deschide in creere fisierul misc.dat\n");
        getch();
        exit(1);
        }
for(; ;) {
        for(i = 0; i < 6 ; i++)  {  /* se umple zona tampon utilizata ptr. fisier */
                if(( n = cit(&a)) == EOF)
                        break;
                buf.a[i] = a;
                }
        /* se scrie continutul zonei tampon in fisierul misc.dat */
        if(i != 0)
                if( i != fwrite(buf.zt,sizeof(ARTICOL),i,pf))  {
                        printf(" Eroare de scriere in fisier \n");
                        getch();
                        exit(1);
                        }
        if(n == EOF)
                break;
        }
fclose(pf);
pf = fopen("misc.dat","rb");
afis(pf);
getch();
fclose(pf);
}

/* citeste datele din intrarea standard si le scrie in structura al carei pointer este str */
```


Scanned with OKEN Scanner

```
int cit(ARTICOL *str)
{
float x;
clrscr();
printf("\n Introduceti datele de intrare.\n ATENTIE iesirea se face cu tip"
       " document = $\n");
printf(" Tipul documentului,maxim 4 caractere - I sau E sau bm sau bmc: ");
scanf("%s",str->tip_doc);
if(strcmp(str->tip_doc,"$") == 0)
          return EOF;
printf(" Denumire (maxim 30 carac.): ");
getchar();
gets(str->den);
printf(" Unitatea de masura (max.3 caractere): ");
scanf("%s",str->um);
printf(" Codul (nr.intreg cu maxim 9 cifre): ");
scanf("%ld",&str->cod);
printf(" Pretul (nr. flotant):              ");
scanf("%f",&x);
str->pret = x;
printf(" Cantitatea (nr. flotant):          ");
scanf("%f",&x);
str->cant = x;
getch();
return 1;
}

void afis(FILE *pf)
{
ARTICOL a;
int i;
unsigned n;
clrscr();
printf("%45s\n","I N T R A R I");
printf("\n\n|----------------------------------------------------------------------|\n");
printf("|tip doc.|    Denumire material      |U.M.| Cod mat.|    Pret    | Cantitate|\n");
printf("|----------------------------------------------------------------------|\n");
while((n = fread(buf.zt,sizeof(ARTICOL),6,pf)) > 0)
          for(i = 0; i < n ; i++)  {
                    a = buf.a[i];
```


Scanned with OKEN Scanner

```
                printf("|%-5s  |%-29s| %-3s|%9ld|"
                  "  %10.2f|%10.2f|\n",a.tip_doc,a.den,a.um,a.cod,a.pret,a.cant);
                }
printf("|-------------------------------------------------------------------------|\n");
}
```

B23) Problema este o aplicație pentru prelucrarea pe șiruri de caractere a fișierelor. Dar în loc de a utiliza funcțiile standard fgets și fputs, se vor face variante proprii: fgetsp și fputsp.

Se cere copierea unui rînd dintr-un fișier al cărui nume este argument în linia de comandă într-un alt fișier, al cărui nume este tot argument în linia de comandă. Dacă în linia de comandă nu sunt argumente, se copie o singură linie din intrarea standard în ieșirea standard. Dacă în linia de comandă este un singur nume de fișier, se semnalează eroare.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

#define LMAX 80

char *fgetsp(char *,int,register FILE *);
void  fputsp(char *,register FILE *);
void filecopy(FILE *);

void main(int argc,char *argv[])
{
char sir[LMAX];
FILE *fp;
clrscr();
if(argc == 1) {
        printf("\nNu-s argumente in linia de comanda:\n");
        printf("Introduceti o singura linie:\n\n");
        fgetsp(sir,LMAX,stdin);
        fputsp(sir,stdout);
        printf("\nApasati o tasta... ");
        getch();
        exit(1);
        }
else {
```


Scanned with OKEN Scanner

```
if(argc == 2)
        printf("\n\n In linia de comanda este un singur argument\n"
              " Trebuiesc doua!!\n");
while(--argc > 1)
        if((fp = fopen(*++argv,"r")) == NULL) {
              printf("\n Fisierul de intrare %s  nu se poate"
                    " deschide\n",*argv);
              break;
              }
        else {
              printf("\n Primul fisier %s  :\n\n",*argv);
              filecopy(fp);
              fclose(fp);
              fp = fopen(*argv,"r");
              fgetsp(sir,LMAX,fp);
              fclose(fp);
              if((fp = fopen(*++argv,"a")) == NULL) {
                    printf("\n\n Fisierul %s nu se poate"
                          " deschide\n",*argv);
                    break;
                    }
              else {
                    fclose(fp);
                    fp = fopen(*argv,"r");
                    printf("\n\n Al doilea fisier %s :\n\n",*argv);
                    filecopy(fp);
                    fclose(fp);
                    fp = fopen(*argv,"a");
                    fputsp(sir,fp);
                    fclose(fp);
                    printf("\n\n Al doilea fisier %s  ,dupa"
                          " adaugare:\n\n",*argv);
                    fp = fopen(*argv,"r");
                    filecopy(fp);
                    fclose(fp);
                    }
        }
}
getch();
}
```


Scanned with OKEN Scanner

```
char *fgetsp(char *s,int n,register FILE *fp)
{
register int c;
register char *cs;
cs = s;
while(--n > 0 && (c = getc(fp)) != EOF)
        if((*cs++ = c) == '\n')
                break;
*cs = '\0';
return((c == EOF && cs == s) ? NULL : s);
}

void fputsp(char *s,register FILE *fp)
{
register int c;
while(c = *s++)
        putc(c,fp);
}

void filecopy(FILE *fp)
{
int c;
while((c = getc(fp)) != EOF)
        putc(c,stdout);
}
```

B24) Să se scrie un program pentru copierea a unu, două sau mai multe fişiere în ieşirea standard. Numele fişierelor sunt argumente în linia de comandă. Dacă aceste argumente lipsesc, se copie intrarea standard în ieşirea standard. Se va folosi prelucrarea fişierelor la nivel superior. Se va testa o funcţie de copiere a unui fişier în ieşirea standard.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

void filecopy(FILE *);

void main(int argc,char *argv[])
{
```


Scanned with OKEN Scanner

```
FILE *fp;
clrscr();
if(argc == 1) {  /* nu-s argumente: copie fisierul standard de intrare */
        printf("\n se copie fisierul standard de intrare:\n\n");
        filecopy(stdin);
        exit(1);
        }
else {
        printf("Fisierele concatenate:\n\n");
        while( --argc > 0 )
                if(( fp = fopen(*++argv,"r")) == NULL)  {
                        printf("Nu se poate deschide fisierul  %s\n",*argv);
                        break;
                        }
                else  {
                        filecopy(fp);
                        fclose(fp);
                        }
                }
getch();
}

void filecopy(FILE *fp)  /* copie fisierul in iesirea standard */
        {
        int c;
        while(( c = getc(fp)) != EOF)
                putc(c,stdout);
        }
```

B25) Să se scrie un program pentru concatenarea în ieşirea standard a două sau mai multe fişiere ale căror nume sunt date ca argumente în linia de comandă.

Dacă în linia de comandă nu este nici un argument, se copie intrarea standard în ieşirea standard.

Se va scrie o funcţie pentru a copia un fişier în ieşirea standard.

Se va folosi prelucrarea fişierelor la nivel coborît.

```
#include <stdio.h>
#include <conio.h>
#include <process.h>
#include <io.h>
```


Scanned with OKEN Scanner

```
#include <fcntl.h>

#define BUF 70

void fcopy(int df);

void main(int argc,char *argv[])
{
int df,i;
clrscr();
if(argc == 1) {
        printf(" Pentru ca nu sunt argumente in linia de comanda,intrarea"
              " standard va fi copiata in iesirea standard !!");
        printf("\n ATENTIE: Iesirea din bucla de citire se face cu ^Z\n");
        printf("\n Introduceti randurile de la tastatura!\n");
        fcopy(0);
        exit(1);
        }
else {
        printf("Fisierele concatenate:\n\n");
        for(i = 1; --argc>0 ; ++i) {
                df = open(argv[i],O_RDWR);
                if(df < 0) {
                        printf("\n Fisierul %s nu poate fi deschis!\n",argv[i]);
                        getch();
                        exit(1);
                        }
                fcopy(df);
                close(df);
                }
        }
getch();
}

void fcopy(int df)
{
char bf[BUF];
int lung;
while((lung = read(df,bf,BUF)) > 0)
        write(1,bf,lung);
}
```


Scanned with OKEN Scanner

B26) Să se citească un şir de numere reale şi să se scrie aceste numere în două fişiere fis1.dat şi fis2.dat astfel : citirile de ordin impar să fie scrise în fis1.dat iar numerele din citirile de ordin par în fis2.dat. Să se afişeze apoi conţinutul fişierelor obţinute. Se va folosi prelucrarea fişierelor la nivel coborît.

```
#include <stdio.h>
#include <\sys\stat.h>
#include <io.h>
#include <fcntl.h>
#include <conio.h>
#include <stdlib.h>

void main(void)
{
union unr {
        float nr;
        char tnr[sizeof(float)];
        }nrcit;
int df1,df2,i,j;
float x;
_fmode = O_BINARY;
clrscr();
/* se deschid fisierele fis1.dat si fis2.dat ptr. creare */
if((df1 = creat("fis1.dat",S_IWRITE )) == -1)  {
        printf("\n nu se poate deschide fisierul fis1.dat! \n");
        getch();
        exit(1);
        }
if((df2 = creat("fis2.dat",S_IWRITE )) == -1)   {
        printf("\n nu se poate deschide fisierul fis2.dat! \n");
        getch();
        exit(1);
        }
/* se citeste sirul de numere si se pastreaza in cele doua fisiere */
printf("\n Introduceti mai multe numere reale:\n");
i=1;
j=1;
while( i == 1 ) {
        printf(" citirea cu nr. %2d :  ",j);
```


Scanned with OKEN Scanner

```
        i = scanf("%f",&x);
        if(i == EOF)
                break;
        nrcit.nr = x;
        if(j % 2 == 1)  { /* se scrie in fis1.dat  */
                if((write(df1,nrcit.tnr,sizeof(float))) != sizeof(float))  {
                        printf("\n eroare de scriere in fisierul fis1.dat\n");
                        getch();
                        exit(1);
                        }
                }
        else            /* se scrie in fisierul fis2.dat */
                if((write(df2,nrcit.tnr,sizeof(float))) != sizeof(float))  {
                        printf("\n eroare de scriere in fisierul fis2.dat\n");
                        getch();
                        exit(1);
                        }
        j++;
}

if(close(df1) < 0 )  {
        printf("\n eroare la inchiderea fisierului fis1.dat\n");
        getch();
        exit(1);
        }
if(close(df2) < 0)  {
        printf("\n eroare la inchiderea fisierului fis2.dat\n");
        getch();
        exit(1);
        }
if(( df1 = open("fis1.dat",O_RDONLY)) == -1)  {
        printf("\n nu se poate deschide fis1.dat in citire\n");
        getch();
        exit(1);
        }
printf("\n\n Continutul fisierului fis1.dat\n");
j = 1;
while(( i = read(df1,nrcit.tnr,sizeof(float))) > 0) {
        printf(" numarul %2d este  %10.2f\n",j,nrcit.nr);
        j += 2;
        }
```


Scanned with OKEN Scanner

```
if( i < 0 )  {
        printf("\n Eroare la citire fis1.dat\n");
        getch();
        exit(1);
        }
close(df1);
if(( df2 = open("fis2.dat",O_RDONLY)) == -1)  {
        printf("\n nu se poate deschide fis2.dat in citire\n");
        getch();
        exit(1);
        }
printf("\n\n Continutul fisierului fis2.dat\n");
j = 2;
while(( i = read(df2,nrcit.tnr,sizeof(float))) > 0) {
        printf(" numarul %2d este  %10.2f\n",j,nrcit.nr);
        j += 2;
        }
if( i < 0 )  {
        printf("\n Eroare la citire fis2.dat\n");
        getch();
        exit(1);
        }
close(df2);
getch();
}
```

B27) Să se scrie un program care realizează urmatoarele:

- desenează pe ecran un dreptunghi şi un cerc pe care le umple, pe rînd cu toate modurile de umplere standardizate şi cu un mod de umplere definit de utilizator;
- desenează patru cercuri care nu se intersectează şi umple spaţiul dintre ele cu SOLID_FILE.
- comută SOLID_FILE pe cercuri, lăsînd spaţiul dintre ele la culoarea fondului.

```
#include <graphics.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <conio.h>
```


Scanned with OKEN Scanner

```
/* numele stilurilor de umplere */
char *nume_stil[] = {    "EMPTY_FILL",
                         "SOLID_FILL",
                         "LINE_FILL",
                         "LTSLASH_FILL",
                         "SLASH_FILL",
                         "BKSLASH_FILL",
                         "LTBKSLASH_FILL",
                         "HATCH_FILL",
                         "XHATCH_FILL",
                         "INTERLEAVE_FILL",
                         "WIDE_DOT_FILL",
                         "CLOSE_DOT_FILL",
                         "USER_FILL"
                    };
char userpattern[8] = {0xDD,0xDD,0xDD,0xDD,0xDD,0xDD,0xDD,0xDD};

int main(void)
{
 int gdriv = DETECT, gmod, coder;
 int stil, medx, medy,xmax,ymax;
 char sir_stil[40];
 initgraph(&gdriv, &gmod, "e:\\tc\\bgi");  /* initializarea modului  graphic  */
 coder = graphresult();   /* rezultatul initializarii */
 if (coder != grOk)       /*  eroare de initializare  */
         {
                         printf("Eroare grafica: %s\n", grapherrormsg(coder));
                         printf("Apasati orice tasta ptr. a iesi:");
                         getch();
                         exit(1); /* iesire din eroare */
         }

medx = getmaxx() / 2;
medy = getmaxy() / 2;
for(stil = EMPTY_FILL; stil < USER_FILL; stil++)
         {
                 setfillstyle(stil,getmaxcolor());  /* selectarea stilului de umplere */
                 strcpy(sir_stil,nume_stil[stil]);
                                 /* conversia stilului in sir de caractere */
                 bar3d(1,0,200,150,1,1);
```


Scanned with OKEN Scanner

```
                pieslice(medx,medy,0,360,100);
                outtextxy(medx+120, medy+120,sir_stil);
                getch();
                cleardevice();
        }
setfillstyle(USER_FILL,getmaxcolor());
setfillpattern(userpattern,getmaxcolor());
bar3d(1,0,200,150,1,1);
pieslice(medx,medy,0,360,100);
outtextxy(medx+120, medy+120, "USER_PATERN");
getch();
cleardevice();

xmax = getmaxx();
ymax = getmaxy();
setcolor(getmaxcolor()); /* selectarea culorii */
setfillstyle(SOLID_FILL, getmaxcolor()); /* culoarea de umplere */
rectangle(0, 0, xmax, ymax); /*chenarul ecranului */
circle(xmax / 4, ymax /2, 50);
circle(xmax / 2, 80, 100);
circle(xmax-120, ymax-150, 75);
circle(30, ymax-25, 25);
floodfill(2, 2, getmaxcolor());
getch();
cleardevice();
circle(xmax / 4, ymax /2, 50);
circle(xmax / 2, 80, 100);
circle(xmax-120, ymax-150, 75);
circle(30, ymax-25, 25);
floodfill(xmax/4, ymax/2, getmaxcolor());
floodfill(xmax/2,80 , getmaxcolor());
floodfill(xmax-120,ymax-150 , getmaxcolor());
floodfill(30,ymax-25 , getmaxcolor());
getch();
cleardevice();
closegraph();
return 0;
}
```


Scanned with OKEN Scanner

B28) Să se scrie un program prin care se încadrează doua texte în dreptunghiuri verticale şi un text într-un dreptunghi orizontal. Textele vor fi scrise cu fonturi diferite.

```
#include <graphics.h>
#include <conio.h>

void cadru_text(int st,int sus,int dr,int jos,int font,char *sir);
void cadru_vert_text(int st,int sus,int dr,int jos,int font,char * sir);

void main()
{
int gmod,gdriv = DETECT;
char *sir = "Grafica in C++";
initgraph(&gdriv,&gmod,"E:\\tc\\bgi");
cadru_vert_text(0,0,200,getmaxy(),TRIPLEX_FONT,sir);
cadru_text(220,20,500,110,GOTHIC_FONT,sir);
cadru_vert_text(520,0,getmaxx(),getmaxy(),SANS_SERIF_FONT,sir);
getch();
closegraph();
}

void cadru_text(int st,int sus,int dr,int jos,int font,char *sir)
{
setusercharsize(1,1,1,1);
settextstyle(font,HORIZ_DIR,USER_CHAR_SIZE);
setusercharsize(dr-st,textwidth(sir),jos-sus,textheight(sir)*3/2);
settextjustify(CENTER_TEXT,CENTER_TEXT);
setfillstyle(SOLID_FILL,getbkcolor());
bar3d(st,sus,dr,jos,0,0);
outtextxy((dr+st)/2,(jos-sus)/2,sir);
}

void cadru_vert_text(int st,int sus,int dr,int jos,int font,char *sir)
{
setusercharsize(1,1,1,1);
settextstyle(font,VERT_DIR,USER_CHAR_SIZE);
setusercharsize(jos-sus,textwidth(sir),dr-st,textheight(sir)*3/2);
settextjustify(CENTER_TEXT,CENTER_TEXT);
setfillstyle(SOLID_FILL,getbkcolor());
bar3d(st,sus,dr,jos,0,0);
```


Scanned with OKEN Scanner

```
outtextxy((dr+st)/2,(jos-sus)/2,sir);
}
```

B29) Să se scrie un program care desenează un dreptunghi pe ecran şi-l umple pe rînd cu toate stilurile de umplere. Pentru fiecare stil în parte, va fi scris numele acelui stil. Trecerea de la un stil la altul se face prin apăsarea unei taste.

```
#include <graphics.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <conio.h>

/* tabel de pointeri pe numele stilurilor de umplere */
char *fname[] = { "EMPTY_FILL",
                  "SOLID_FILL",
                  "LINE_FILL",
                  "LTSLASH_FILL",
                  "SLASH_FILL",
                  "BKSLASH_FILL",
                  "LTBKSLASH_FILL",
                  "HATCH_FILL",
                  "XHATCH_FILL",
                  "INTERLEAVE_FILL",
                  "WIDE_DOT_FILL",
                  "CLOSE_DOT_FILL",
                  "USER_FILL"
                };

void main(void)
{
  int gdriver = DETECT, gmode, errorcode;
  int style, midx, midy;
  char stylestr[40];
  initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
  errorcode = graphresult();
  if (errorcode != grOk)
  {
        printf("Eroare de initializare: %s\n", grapherrormsg(errorcode));
        printf("Apasati orice tasta .");
    getch();
```


Scanned with OKEN Scanner

```
        exit(1); /* terminat in caz de eroare */
}

midx = getmaxx() / 2;
midy = getmaxy() / 2;
for (style = EMPTY_FILL; style < USER_FILL; style++)
{
        /* selectarea stilului de umplere */
        setfillstyle(style, getmaxcolor());
        /* selectarea numelui stilului */
        strcpy(stylestr, fname[style]);
        /* dreptunghiul care se va umple cu stilul selectat */
        bar3d(0, 0, midx-10, midy, 0, 0);
        /* scrierea numelui stilului de umplere */
        outtextxy(midx, midy, stylestr);
        /* se asteapta o tasta */
        getch();
        cleardevice();
  }
  getch();
  closegraph();
}
```

B30) Să se deseneze un semicerc şi diametrul pe care se sprijină.

```
#include <graphics.h>
#include <conio.h>

void main(void)
{
struct arccoordstype arccord;
int gmod,gdriv = DETECT;
initgraph(&gdriv,&gmod,"e:\\tc\\bgi");
arc(getmaxx()/2,getmaxy()/2,0,180,getmaxx()/4);
getarccoords(&arccord);
line(arccord.xstart,arccord.ystart,arccord.xend,arccord.yend);
getch();
closegraph();
}
```


Scanned with OKEN Scanner

B31) Să se scrie un program care desenează aleator 1000 de pixeli pe ecran, apoi îi şterge ponderat (din doi in doi). După ştergerea ponderată, se realizează o ştergere linie cu linie.

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <stdlib.h>
#include <dos.h>

void main(void)
{
int gdriv = DETECT,gmod,xmax,ymax,i,x,y,color,cl;
initgraph(&gdriv,&gmod,"E:\\tc\\bgi");
xmax = getmaxx();
ymax = getmaxy();
color = getmaxcolor();
outtextxy(0,0,"Asteptati: merge incet ...");
randomize();
for(i = 0; i < 1000; ++i)  {
        x = random(xmax + 1);
        y = random(ymax + 1);
        cl = random(color + 1);
        delay(5);
        putpixel(x,y,cl);
        }
outtextxy(0,10,"Pentru a sterge ponderat,apasati o tasta...");
getch();
i = 0;
for(y = 0; y < ymax; ++y)
        for(x = 0; x < xmax; ++x)
                if(getpixel(x,y) != BLACK)
                        if(i%2 == 0)  {
                                putpixel(x,y,BLACK);
                                ++i;
                                }
outtextxy(0,20,"Pentru a sterge linie cu linie,apasati o tasta...");
getch();
for(y = 0; y < ymax; ++y)
```


Scanned with OKEN Scanner

```
        for(x = 0; x < xmax; ++x)
                if(getpixel(x,y) != BLACK)
                                putpixel(x,y,BLACK);
outtextxy(0,10,"Pentru a termina,apasati orice tasta...");
getch();
closegraph();
}
```

B32) Să se scrie un program care realizează următoarele:
- scrie un text, centrat pe mijlocul ecranului;
- la fiecare apăsare a tastei 'M' să se mărească textul cu un increment prestabilit;
- la fiecare apăsare a tastei 'm' să se micşoreze textul cu acelaşi increment;
- la apăsarea tastei 'd' se schimbă direcţia textului: din orizontal în vertical şi invers;
- la apăsarea tastei '$' programul se termină.

```
#include <conio.h>
#include <graphics.h>
#define INC 20

int multx,multy,divx,divy;
int x,y,lung,inalt;
int dir,ndir;
char *sir;

void initstil(void);
void afisare(void);
int dimensiune(void);
void sterge(void);

void main(void)
{
int ch;
int gdriv = DETECT,gmod;
multx = multy = divx = divy = 100;
dir = ndir = HORIZ_DIR;
sir = "Turbo C++";
initgraph(&gdriv,&gmod,"E:\\tc\\bgi");
outtextxy(0,0,"M - marire  m - micsorare");
outtextxy(0,10,"d - schimbare de directie");
outtextxy(0,20,"$ - terminare program");
```


Scanned with OKEN Scanner

```
initstil();
do {
        afisare();
        ch = dimensiune();
        sterge();
        } while(ch != '$');
closegraph();
}

void initstil(void)
{
setfillstyle(SOLID_FILL,BLACK);
x = getmaxx()/2;
y = getmaxy()/2;
settextjustify(CENTER_TEXT,CENTER_TEXT);
}

void afisare(void)
{
setusercharsize(multx,divx,multy,divy);
settextstyle(GOTHIC_FONT,dir,USER_CHAR_SIZE);
outtextxy(x,y,sir);
}

int dimensiune(void)
{
int c = getch();
if(c == 'M')  {
        if(divx > INC) {
                divx -=INC;
                multx += INC;
                }
        if(divy > INC)  {
                divy -= INC;
                multy += INC;
                }
        }
else
        if( c == 'm')  {
                if(multx > INC) {
                        multx -= INC;
```


Scanned with OKEN Scanner

```
                    divx += INC;
                    }
                if(multy > INC) {
                    multy -= INC;
                    divy += INC;
                    }
                }
        else
                if(c == 'd')
                  ndir = (ndir == HORIZ_DIR) ? VERT_DIR : HORIZ_DIR;
return c;
}

void sterge(void)
{
lung = textwidth(sir)/2;
inalt = textheight(sir)/2;
if(dir == HORIZ_DIR)  {
        if(lung > x)
                lung = x+1;
        if(inalt > y)
                inalt = y+1;
        bar(x - lung, y - inalt, x + lung, y + inalt);
        }
else  {
        if(inalt > x)
                inalt = x+1;
        if(lung > y)
                lung = y+1;
        bar(x - inalt, y - lung, x + inalt + 10, y + lung);
}
dir = ndir;
}
```

B33) Să se scrie un program de desenare a următoarelor figuri: un sector de cerc, un cerc, un patrulater, o elipsă, o bară şi o bară în trei dimensiuni. Primele trei figuri vor fi desenate în jumătatea de sus a ecranului, iar următoarele trei în jumătatea de jos. Toate figurile vor fi haşurate şi inscripţionate. Calculul coordonatelor se va face în funcţie de coordonatele maxime ale ecranului.

Scanned with OKEN Scanner

```
#include <graphics.h>
#include <conio.h>

void main(void)
{
int gdriv = DETECT,gmod,xmax,ymax;
int patr[10];
initgraph(&gdriv,&gmod,"E:\\tc\\bgi");
xmax = getmaxx();
ymax = getmaxy();
setfillstyle(LTBKSLASH_FILL,WHITE);
pieslice(xmax/6,ymax/3,0,130,50);
outtextxy(xmax/6 - textwidth("sector")/2,20,"sector");
pieslice(xmax/2,ymax/3,0,360,50);
outtextxy(xmax/2 - textwidth("cerc umplut")/2,20,"cerc umplut");
patr[0] = 5*xmax/6 - 20;
patr[1] = ymax/3;
patr[2] = 5*xmax/6 - 30;
patr[3] = ymax/3 + 30;
patr[4] = 5*xmax/6 + 40;
patr[5] = ymax/3 + 25;
patr[6] = 5*xmax/6 + 10;
patr[7] = ymax/3 - 30;
patr[8] = patr[0];
patr[9] = patr[1];
fillpoly(5,patr);
outtextxy(5*xmax/6 - textwidth("patrulater umplut")/2,20,"patrulater umplut");
sector(xmax/6,3*ymax/4,0,360,70,20);
outtextxy(xmax/6 - textwidth("elipsa umpluta")/2,ymax - 10,"elipsa umpluta");
bar(xmax/2 + 20,3*ymax/4 - 30,xmax/2 + 20, 3*ymax/4 + 30);
outtextxy(xmax/2 - textwidth("bara")/2,ymax - 10,"bara");
bar3d(5*xmax/6 - 20, 3*ymax/4 - 30,5*xmax/6 + 20,3*ymax/4 + 30,10,1);
outtextxy(5*xmax/6 - textwidth("bara in 3d")/2,ymax - 10,"bara in 3d");
getch();
closegraph();
}
```


Scanned with OKEN Scanner

Referințe bibliografice pentru programarea în C

[1] Ammeraal, Lendert, *C For Programmers (Second Edition)*, John Wiley & Sons, 1991

[2] Dellanoy, Claudes, *Exercises en Turbo C*, Eyrolles, Paris, 1991

[3] Horrowitz, Ellis, Sartaj, Sahni, *Fundamentals of Data Structures in Pascal*, Computer Science Press, Rockville, 1984

[4] Kelly-Bootle, Stan, *Mastering Turbo C (Second Edition)*, Sybex, 1990

[5] Livovschi, Leon, Georgescu, Horia, *Sinteza și analiza algoritmilor*, Editura Științifică și Enciclopedică, București, 1986

[6] ***, *Borland C++ Version 2.0 Library Reference*, Borland International, 1991

[7] ***, *Turbo C++ Second Edition Programmer's Guide*, Borland International, 1991

[8] ***, *Turbo C++ Second Edition User's Guide*, Borland International, 1991

Scanned with OKEN Scanner

Capitolul II : Baze de date - dBase IV

Exercițiul 2.2 - 3.1

2.2.a) Scrieți un program care creează următoarele tabele și indexurile corespunzătoare cheilor primare (prima coloană din fiecare):

CLIENTI

nr_client	NUMERIC(8,0)	
denumire	CHAR(15)	
tip	CHAR(3)	
localitate	CHAR(10)	
judet	CHAR(2)	
privat	LOGICAL	

AGENTI

nr_agent	NUMERIC(4,0)	
nume	CHAR(10)	
prenume	CHAR(10)	
oras	CHAR(10)	
judet	CHAR(2)	&& evident , abrevierea
angajare	DATE	
nastere	DATE	
sex	CHAR(1)	
cas	LOGICAL	&& căsătorit - ignorăm divorțat etc.
sal_baza	NUMERIC(6,0)	

TRANZACT && tranzacții

nr_tranz	NUMERIC(6,0)	
nr_client	NUMERIC(8,0)	
nr_agent	NUMERIC(4,0)	
cod_produs	NUMERIC(6,0)	
cant	NUMERIC(4,0)	&& cantitate
data_tranz	DATE	
nr_fact	NUMERIC(6,0)	
tip	CHAR(1)	&& I - intrări , E - ieșiri

Scanned with OKEN Scanner

- simplificat: o tranzacţie se referă la o singură marfă:

```
FACTURI
        nr_fact         NUMERIC(6,0)
        nr_tranz        NUMERIC(6,0)
        data_em         DATE                    && data emiterii
        ach             LOGICAL                 && achitată sau nu
        data_ach        DATE
        valoare         NUMERIC(10,0)

MARFURI
        cod_produs      NUMERIC(6,0)
        denumire        CHAR(12)
        tip_model       CHAR(10)
        un_mas          CHAR(6)
        pret_un         NUMERIC(8,0)  && presupunem preţurile constante în
                                      && timp - indiferent de furnizor
        cantitate       NUMERIC(8,0)  && stoc
```

3.1) Refaceţi procedura creată la lucrarea 2. adăugîndu-i şi încărcarea datelor din fişierele CL.DBF, AG.DBF, TR.DBF, FA.DBF, MA.DBF respectiv în tabelele CLIENTI, ... , MARFURI.

```
PROCEDURE 2_2                   && atenţie la extensia fişierului: .PRS !
CREATE DATABASE d:\zzzz ;
                                && sau orice alt director adecvat
CREATE TABLE clienti (
        nr_client       NUMERIC(8,0) ,
        denumire        CHAR(15) ,
        tip             CHAR(3) ,
        localitate      CHAR(10) ,
        judet           CHAR(2) ,
        privat          LOGICAL ) ;
CREATE UNIQUE INDEX ind_cl ON clienti ( nr_client ) ;
CREATE TABLE agenti (
        nr_agent        NUMERIC(4,0) ,
        nume            CHAR(10) ,
        prenume         CHAR(10) ,
        oras            CHAR(10) ,
        judet           CHAR(2) ,
        angajare        DATE ,
```


Scanned with OKEN Scanner

```
        nastere         DATE ,
        sex             CHAR(1) ,
        cas             LOGICAL ,
        sal_baza        NUMERIC(6,0) ) ;
CREATE UNIQUE INDEX ind_ag ON agenti ( nr_agent ) ;
CREATE TABLE tranzact (
        nr_tranz        NUMERIC(6,0) ,
        nr_client       NUMERIC(8,0) ,
        nr_agent        NUMERIC(4,0) ,
        cod_produs      NUMERIC(6,0) ,
        cant            NUMERIC(4,0) ,
        data_tranz      DATE ,
        nr_fact         NUMERIC(6,0) ,
        tip             CHAR(1) ) ;
CREATE UNIQUE INDEX ind_tr ON tranzact ( nr_tranz ) ;
CREATE TABLE facturi (
        nr_fact         NUMERIC(6,0) ,
        nr_tranz        NUMERIC(6,0) ,
        data_em         DATE ,
        ach             LOGICAL ,
        data_ach        DATE ,
        valoare         NUMERIC(10,0) ) ;
CREATE UNIQUE INDEX ind_fa ON facturi ( nr_fact ) ;
CREATE TABLE marfuri (
        cod_produs      NUMERIC(6,0) ,
        denumire        CHAR(12) ,
        tip_model       CHAR(10) ,
        un_mas CHAR(6) ,
        pret_un NUMERIC(8,0) ,
        cantitate       NUMERIC(8,0) ) ;
CREATE UNIQUE INDEX ind_ma ON marfuri ( cod_produs ) ;

*  Se presupune că fişierele de date se află în directorul curent (a nu se confunda cu
* directorul corespunzător bazei de date SQL).

LOAD DATA FROM cl  INTO TABLE clienti ;
LOAD DATA FROM ag  INTO TABLE agenti ;
LOAD DATA FROM tr  INTO TABLE tranzact ;
LOAD DATA FROM fa  INTO TABLE facturi ;
LOAD DATA FROM ma INTO TABLE marfuri ;
RETURN
```


Scanned with OKEN Scanner

Atenție: ordinea operațiilor nu este indiferentă - trebuie mai întîi să se creeze indexul cu clauza *UNIQUE* și abia după aceea să se încarce datele în tabele (se va refuza orice încărcare / actualizare ce nu respectă unicitatea valorilor vizate).

Dacă s-ar încărca datele înaintea creării indexului, duplicarea vreunei valori în coloana corespunzătoare clauzei *UNIQUE* ar împiedica însăși crearea indexului.

Exercițiul 3.3

Adăugați în tabelul AGENTI o coloană numerică PRIME; completați-i valorile după cum urmează: 100 000 pentru cei din orașul Iași, 75 000 pentru cei din restul județului, 60 000 pentru ceilalți.

```
PROCEDURE 3_3
ALTER TABLE agenti ADD ( prime  NUMERIC(7,0) ) ;
SELECT nr_agent , oras , judet , prime FROM agenti ;
UPDATE agenti SET prime = 100000 WHERE judet='IS' AND oras='IASI' ;
UPDATE agenti SET prime =  75000 WHERE judet='IS' AND oras<>'IASI' ;
UPDATE agenti SET prime =  60000 WHERE judet<>'IS' ;
SELECT nr_agent , oras , judet , prime FROM agenti ;
RETURN
```

Dacă nu folosim vreuna din funcțiile de conversie (*UPPER()* / *LOWER()*), trebuie să indicăm șirurile de caractere (de exemplu 'IASI') exact așa cum apar în baza de date!

Exercițiul 3.4

Ștergeți din tabelul MARFURI liniile cu cantitatea sub 10 ori prețul unitar sub 150.

Atenție: "ori" este sinonim - și - cu "sau", deci nu trebuie interpretat ca 10 * prețul unitar. Oricum, această "înțelegere" ar trebui să fie anulată imediat ce se ajunge cu citirul la partea cu "sub 150", în măsura în care propoziția nu este deja prea lungă.

```
PROCEDURE 3_4
```


Scanned with OKEN Scanner

```
SELECT cod_produs , cantitate , pret_un FROM marfuri ;
DELETE FROM marfuri WHERE cantitate < 10 OR pret_un <150 ;
SELECT cod_produs , cantitate , pret_un FROM marfuri ;
RETURN
```

Exercițiul 3.5

Adăugați în tabelul FACTURI o coloană TVA; completați-i toate valorile cu 18 % din coloana VALOARE.

```
PROCEDURE 3_5
ALTER TABLE facturi ADD ( tva NUMERIC(12.0) ) ;
UPDATE facturi SET tva = 0.18 * valoare WHERE .T. ;
SELECT nr_fact , valoare , tva FROM facturi ;
RETURN
```

Exercițiul 3.6

Avînd exemplul din textul lucrării, determinați pentru fiecare tabel toate cazurile de nerespectare a integrității de referință (valoare a unei chei externe inexistentă în tabelul unde este cheie primară).

Cîte asemenea cazuri ați găsit pentru fiecare tabel (se prezintă varianta cu clauza *COUNT* - prezentată într-o lucrare ulterioară)?

```
PROCEDURE 3_6
CLEAR
SET TALK off
cl = 0
ag = 0
cp = 0
fa = 0
tr = 0
SELECT COUNT(*) INTO cl FROM tranzact
        WHERE nr_client NOT IN (
                SELECT nr_client FROM clienti ) :
SELECT COUNT(*) INTO ag FROM tranzact
        WHERE nr_agent NOT IN (
                SELECT nr_agent FROM agenti ) :
SELECT COUNT(*) INTO cp FROM tranzact
```


Scanned with OKEN Scanner

```
        WHERE cod_produs NOT IN (
                SELECT cod_produs FROM marfuri ) :
SELECT COUNT(*) INTO fa FROM tranzact
        WHERE nr_fact NOT IN (
                SELECT nr_fact FROM facturi ) :
SELECT COUNT(*) INTO tr FROM facturi
        WHERE nr_tranz NOT IN (
                SELECT nr_tranz FROM tranzact ) :
WAIT "TRANZACT : clienti inexistenti   - " + STR(cl,3) + " caz(uri) ..."
SELECT nr_tranz , nr_client FROM tranzact
        WHERE nr_client NOT IN (
                SELECT nr_client FROM clienti ) :
WAIT "TRANZACT : agenti inexistenti    - " + STR(ag,3) + " caz(uri) ..."
SELECT nr_tranz , nr_agent FROM tranzact
        WHERE nr_agent NOT IN (
                SELECT nr_agent FROM agenti ) :
WAIT "TRANZACT : marfuri inexistente   - " + STR(cp,3) + " caz(uri) ..."
SELECT nr_tranz , cod_produs FROM tranzact
        WHERE cod_produs NOT IN (
                SELECT cod_produs FROM marfuri ) :
WAIT "TRANZACT : facturi inexistente   - " + STR(fa,3) + " caz(uri) ..."
SELECT nr_tranz , nr_fact  FROM tranzact
        WHERE nr_fact NOT IN (
                SELECT nr_fact FROM facturi ) :
WAIT "FACTURI  : tranzactii inexistente - " + STR(tr,3) + " caz(uri) ..."
SELECT nr_fact , nr_tranz FROM facturi
        WHERE nr_tranz NOT IN (
                SELECT nr_tranz FROM tranzact ) :
SET TALK on
RETURN
```

Exercițiul 3.7

Testați *WHERE - LIKE* cu o variabilă de memorie citită de la tastatură.

```
PROCEDURE 3_7
CLEAR
SET TALK off
SELECT nr_agent , nume , prenume FROM agenti :
ACCEPT "Numele de agent dorit : " TO numeagent
```


Scanned with OKEN Scanner

```
SELECT nr_agent , nume , prenume FROM agenti
        WHERE RTRIM(nume) LIKE numeagent :
* Testați și fără RTRIM(), atît pentru nume "complete" (de exemplu 'IONESCU').
* cît și pentru folosirea wildcard ( 'I%' ).
? "Cu conversia in litere mari :"
SELECT nr_agent , nume FROM agenti
        WHERE RTRIM(UPPER(nume)) LIKE UPPER(numeagent) :
SET TALK on
RETURN
```

Dacă se folosește *INPUT*, ar trebui introdus numele dorit încadrat între apostrofuri sau ghilimele : Numele de agent dorit : 'ION%'

La *ACCEPT*, nu se pun și delimitatorii pentru șiruri de caractere.

Exercițiul 4.3

Care sînt perechile de agenți din același oraș?

```
PROCEDURE 4_3
SELECT a1.nr_agent , a1.nume , a1.oras , a2.nr_agent , a2.nume
        FROM agenti a1 , agenti a2
                WHERE a1.nr_agent < a2.nr_agent
                        AND a1.oras = a2.oras :
RETURN
```

Exercițiul 4.4

Care sînt perechile de clienți din același județ?

```
PROCEDURE 4_4
SELECT c1.denumire , c1.localitate , c1.judet , c2.denumire , c2.localitate
        FROM clienti c1 , clienti c2
                WHERE c1.nr_client < c2.nr_client
                        AND c1.judet = c2.judet :
RETURN
```

- s-ar putea (în sensul că de regulă ar trebui) să se afișeze și numerele de identificare ale clienților:

Scanned with OKEN Scanner

Exercițiul 4.5

Care sînt agenții ce au lucrat numai cu firme private?

- se reformulează problema astfel: care sînt agenții care au lucrat (apar în tabelul TRANZACT) și nu au operat cu firme de stat?

```
PROCEDURE 4_5
SELECT nr_agent , nume , oras FROM agenti
    WHERE nr_agent IN ( SELECT nr_agent FROM tranzact )
        AND nr_agent NOT IN (
            SELECT nr_agent FROM tranzact
                WHERE nr_client IN (
                    SELECT nr_client FROM clienti
                        WHERE privat = .F. ) ) ;
RETURN
```

Exercițiul 4.6

Care sînt agenții care au intermediat aceleași mărfuri (același *cod_produs*; analog s-ar face și pentru aceeași *denumire*)?

```
PROCEDURE 4_6
SELECT a1.nr_agent , a1.nume , a2.nr_agent , a2.nume , denumire
    FROM agenti a1 , agenti a2 , tranzact t1 , tranzact t2 , marfuri
        WHERE a1.nr_agent < a2.nr_agent
            AND a1.nr_agent = t1.nr_agent
            AND a2.nr_agent = t2.nr_agent
            AND t1.cod_produs = t2.cod_produs
            AND t1.cod_produs = marfuri.cod_produs ;
RETURN
```

Exercițiul 4.7

Care sînt clienții cu care plățile s-au făcut într-un singur sens?

```
PROCEDURE 4_7
CLEAR
```


Scanned with OKEN Scanner

```
? "Numai intrări : "
SELECT nr_client , denumire , localitate
        FROM clienti
                WHERE nr_client IN (
                        SELECT nr_client FROM tranzact WHERE tip = 'I' )
                AND nr_client NOT IN (
                        SELECT nr_client FROM tranzact WHERE tip = 'E' ) ;
? "Numai ieşiri : "
SELECT nr_client , denumire , localitate
        FROM clienti
                WHERE nr_client IN (
                        SELECT nr_client FROM tranzact WHERE tip = 'E' )
                AND nr_client NOT IN (
                        SELECT nr_client FROM tranzact WHERE tip = 'I' ) ;
RETURN
```

Exercițiul 4.8

Care sînt perechile de clienți care au furnizat mărfuri identice?

```
PROCEDURE 4_8
SELECT c1.nr_client , c1.denumire , c2.nr_client , c2.denumire , marfuri.denumire
        FROM clienti c1 , clienti c2 , tranzact t1 , tranzact t2 , marfuri
                WHERE c1.nr_client < c2.nr_client
                    AND c1.nr_client = t1.nr_client
                    AND c2.nr_client = t2.nr_client
                    AND t1.cod_produs = t2.cod_produs
                    AND t1.tip = 'I'
                    AND t2.tip = 'I'
                    AND t1.cod_produs = marfuri.cod_produs ;
RETURN
```

Exercițiul 4.9

Care sînt perechile de agenți care au intermediat vînzări de mărfuri identice?

```
PROCEDURE 4_9
SELECT a1.nr_agent , a1.nume , a2.nr_agent , a2.nume , denumire
        FROM agenti a1 , agenti a2 , t    zact t1 , tranzact t2 , marfuri
```


Scanned with OKEN Scanner

```
                WHERE a1.nr_agent < a2.nr_agent
                    AND a1.nr_agent = t1.nr_agent
                    AND a2.nr_agent = t2.nr_agent
                    AND t1.cod_produs = t2.cod_produs
                    AND t1.tip = 'E'
                    AND t2.tip = 'E'
                    AND t1.cod_produs = marfuri.cod_produs ;
RETURN
```

Exercițiul 4.10

Ce mărfuri au un furnizor unic? Dar un "consumator" unic?

```
PROCEDURE 4_10
CLEAR
? "Mărfuri cu un furnizor unic : "
SELECT DISTINCT marfuri.cod_produs, marfuri.denumire,
                clienti.denumire, localitate
        FROM marfuri , clienti , tranzact
                WHERE marfuri.cod_produs NOT IN (
                    SELECT t1.cod_produs
                            FROM clienti c1, clienti c2, tranzact t1, tranzact t2
                                WHERE c1.nr_client < c2.nr_client
                                    AND c1.nr_client = t1.nr_client
                                    AND c2.nr_client = t2.nr_client
                                    AND t1.cod_produs = t2.cod_produs
                                    AND t1.tip = 'I'
                                    AND t2.tip = 'I' )
                AND tranzact.cod_produs = marfuri.cod_produs
                AND tranzact.nr_client = clienti.nr_client
                AND tranzact.tip = 'I' ;
? "Mărfuri cu un cumpărător unic : "
SELECT DISTINCT marfuri.cod_produs, marfuri.denumire,
                clienti.denumire, localitate
        FROM marfuri , clienti , tranzact
                WHERE marfuri.cod_produs NOT IN (
                    SELECT t1.cod_produs
                            FROM clienti c1, clienti c2, tranzact t1, tranzact t2
                                WHERE c1.nr_client < c2.nr_client
                                    AND c1.nr_client = t1.nr_client
```


Scanned with OKEN Scanner

```
                                    AND c2.nr_client = t2.nr_client
                                    AND t1.cod_produs = t2.cod_produs
                                    AND t1.tip = 'E'
                                    AND t2.tip = 'E' )
            AND tranzact.cod_produs = marfuri.cod_produs
            AND tranzact.nr_client = clienti.nr_client
            AND tranzact.tip = 'E' ;
RETURN
```

Exercițiul 4.11

Ce clienți nu și-au achitat facturile și pentru ce mărfuri?

```
PROCEDURE 4_11
SELECT clienti.nr_client, clienti.denumire, marfuri.denumire, valoare
        FROM clienti , tranzact , facturi , marfuri
                WHERE clienti.nr_client = tranzact.nr_client
                    AND tranzact.tip = 'E'
                    AND tranzact.nr_fact = facturi.nr_fact
                    AND ach = .F.
                    AND tranzact.cod_produs = marfuri.cod_produs ;
RETURN
```

Exercițiul 5.1

Scrieți o procedură care fixează și afișează pe rînd data calendaristică în toate formatele posibile.

```
PROCEDURE 5_1
DECLARE variante[11]
variante[1]  = 'american'
variante[2]  = 'ansi'
variante[3]  = 'british'
variante[4]  = 'french'
variante[5]  = 'german'
variante[6]  = 'italian'
variante[7]  = 'japan'
variante[8]  = 'usa'
variante[9]  = 'mdy'
```


Scanned with OKEN Scanner

```
variante[10] = 'dmy'
variante[11] = 'ymd'
CLEAR
SET TALK off
i = 1
DO WHILE i<12
        SET DATE TO &variante[i]
        @i,0 SAY "Data în formatul " + variante[i]
        @i,25 SAY " : " + DTOC(DATE())
        i = i+1
ENDDO
WAIT "Apăsați orice tastă ..."
SET TALK on
RETURN
```

Exercițiul 5.2

Scrieți (și testați) o funcție care returnează un șir de caractere corespunzător datei în limba română: de exemplu:

Simbata 1 ianuarie 1994

Pentru conversia din număr în șir de caractere se va folosi funcția *STR()*.

```
PROCEDURE 5_2
CLEAR
SET TALK off
WAIT "Rezultatul functiei : " + datarom() + " - apasati o tasta..."
SET TALK on
RETURN

FUNCTION datarom
DECLARE zile[7]
zile[1] = 'Duminica'
zile[2] = 'Luni'
zile[3] = 'Marti'
zile[4] = 'Miercuri'
zile[5] = 'Joi'
zile[6] = 'Vineri'
zile[7] = 'Simbata'
DECLARE luni[12]
luni[1]  = 'ianuarie'
```


Scanned with OKEN Scanner

```
luni[2]  = 'februarie'
luni[3]  = 'martie'
luni[4]  = 'aprilie'
luni[5]  = 'mai'
luni[6]  = 'iunie'
luni[7]  = 'iulie'
luni[8]  = 'august'
luni[9]  = 'septembrie'
luni[10] = 'octombrie'
luni[11] = 'noiembrie'
luni[12] = 'decembrie'
sa = DOW(DATE())
zi = DAY(DATE())
lu = MONTH(DATE())
an = YEAR(DATE())
RETURN zile[sa] + STR(zi,3) + " " + luni[lu] + STR(an,5)
```

Exercițiul 5.3

Asemănător cu **5.2**: o funcție care să returneze un șir de 24 de caractere de forma:

Sim Ian 15 12:23:30 1994

```
PROCEDURE 5_3
CLEAR
SET TALK off
ds = datarom1()
@0,0 SAY "Rezultatul functiei : "
@2,0 SAY ds
@4,0 SAY "Lungimea sirului de caractere returnat : " + STR(LEN(ds),3)
SET TALK on
RETURN

FUNCTION datarom1
DECLARE zile[7]
zile[1] = 'Dum'
zile[2] = 'Lun'
zile[3] = 'Mar'
zile[4] = 'Mic'
zile[5] = 'Joi'
```


Scanned with OKEN Scanner

```
zile[6] = 'Vin'
zile[7] = 'Sim'
DECLARE luni[12]
luni[1]  = 'Ian'
luni[2]  = 'Feb'
luni[3]  = 'Mar'
luni[4]  = 'Apr'
luni[5]  = 'Mai'
luni[6]  = 'Iun'
luni[7]  = 'Iul'
luni[8]  = 'Aug'
luni[9]  = 'Sep'
luni[10] = 'Oct'
luni[11] = 'Nov'
luni[12] = 'Dec'
sa = DOW(DATE())
zi = DAY(DATE())
lu = MONTH(DATE())
an = YEAR(DATE())
RETURN zile[sa] + " " + luni[lu] + STR(zi,3) + " " + TIME() + STR(an,5)
```

Exercițiul 5.5

a) Care este valoarea medie a unei tranzacții ?
b) Care sînt agenții bărbați avînd vîrsta strict între 25 și 50 de ani (dacă aveți nevoie de conversia din șir de caractere în număr, folosiți funcția *VAL()*) ?
c) Cine are data nașterii în ziua (și luna) curentă ?
d) Care este vîrsta medie (numai în ani) a agenților, pe localități de domiciliu ?
e) Care este suma facturilor restante de plătit ? Dar de încasat ?
f) Care agent a încheiat cea mai recentă / veche tranzacție ? Cu ce client ?
g) Care agent a încheiat cea mai valoroasă / "ieftină" tranzacție ? Cu ce client ?
h) Care este media salariilor de bază pe sexe și ani de naștere ?

PROCEDURE 5_5

```
* (a)
*       Formularea este poate neinspirată - valoarea apare în tabelul FACTURI.
SELECT AVG(valoare) FROM facturi ;
* (b)
SELECT nr_agent , nume , prenume , nastere FROM agenti
        WHERE sex = 'M'
```


Scanned with OKEN Scanner

```
                AND VAL(DTOS(DATE())) - VAL(DTOS(nastere)) > 250000
                AND VAL(DTOS(DATE())) - VAL(DTOS(nastere)) < 500000 ;
```
* *DTOS(expD)* returnează un şir de caractere de forma *AAAALLZZ*.
```
* (c)
SELECT nr_agent , nume , prenume , nastere FROM agenti
        WHERE MONTH(nastere) = MONTH(DATE())
            AND DAY(nastere) = DAY(DATE()) ;
* (d)
SELECT oras , AVG ( YEAR(DATE()) - YEAR(nastere) )
        FROM agenti
                GROUP BY oras ;
WAIT 'Apasati orice tasta pentru a trece la punctul (e) ...'
* (e)
CLEAR
@0,0 SAY "De plătit : "
SELECT SUM(valoare) FROM facturi
        WHERE ach = .F.
                AND nr_tranz IN (
                        SELECT nr_tranz FROM tranzact WHERE tip = 'I' ) ;
@5+1,0 SAY "De încasat : "
SELECT SUM(valoare) FROM facturi
        WHERE ach = .F.
                AND nr_tranz IN (
                        SELECT nr_tranz FROM tranzact WHERE tip = 'E' ) ;
* (f)
SELECT agenti.nr_agent , nume , denumire , data_tranz
        FROM agenti , clienti , tranzact
                WHERE agenti.nr_agent = tranzact.nr_agent
                    AND clienti.nr_client = tranzact.nr_client
                    AND data_tranz = (
                            SELECT MIN(data_tranz) FROM tranzact ) ;
SELECT agenti.nr_agent , nume , denumire , data_tranz
        FROM agenti , clienti , tranzact
                WHERE agenti.nr_agent = tranzact.nr_agent
                    AND clienti.nr_client = tranzact.nr_client
                    AND data_tranz = (
                            SELECT MAX(data_tranz) FROM tranzact ) ;
* (g)
SELECT agenti.nr_agent , nume , denumire , valoare
        FROM agenti , clienti , tranzact , facturi
                WHERE agenti.nr_agent = tranzact.nr_agent
```


Scanned with OKEN Scanner

```
                AND clienti.nr_client = tranzact.nr_client
                AND tranzact.nr_tranz = facturi.nr_tranz
                AND valoare = ( SELECT MIN(valoare) FROM facturi ) ;
SELECT agenti.nr_agent , nume , denumire , valoare
        FROM agenti , clienti , tranzact , facturi
                WHERE agenti.nr_agent = tranzact.nr_agent
                AND clienti.nr_client = tranzact.nr_client
                AND tranzact.nr_tranz = facturi.nr_tranz
                AND valoare = ( SELECT MAX(valoare) FROM facturi ) ;
* (h)
SELECT sex , YEAR(nastere) , sal_baza
        FROM agenti
                SAVE TO TEMP agetemp ( sex , an_nas , sal_baza ) ;
SELECT sex , an_nas , AVG(sal_baza)
        FROM agetemp
                GROUP BY sex , an_nas ;
RETURN
```

Exercițiul 5.6

Care este media valorilor tranzacțiilor pe clienți? Dar pe tipuri de clienți?

```
PROCEDURE 5_6
SELECT clienti.nr_client , denumire , clienti.tip , valoare
        FROM clienti , tranzact , facturi
                WHERE clienti.nr_client = tranzact.nr_client
                AND tranzact.nr_tranz = facturi.nr_tranz
                SAVE TO TEMP clival ;
CLEAR
@,5.0 SAY "Media pe clienti : "
SELECT denumire , AVG(valoare)
        FROM clival
                GROUP BY denumire ;
@,5+1.0 SAY "Media pe tipuri de clienti : "
SELECT tip , AVG(valoare)
        FROM clival
                GROUP BY tip ;
RETURN
```


Scanned with OKEN Scanner

Exercițiul 5.7

Care este suma facturilor restante de la clienții neprivați? Dar a celor neachitate clienților privați?

```
PROCEDURE 5_7
SELECT clienti.nr_client , privat , valoare
        FROM clienti , tranzact , facturi
                WHERE clienti.nr_client = tranzact.nr_client
                   AND tranzact.nr_fact = facturi.nr_fact
                   AND tranzact.tip = 'E'
                   AND privat = .F.
                SAVE TO TEMP clival1 ;
SELECT clienti.nr_client , privat , valoare
        FROM clienti , tranzact , facturi
                WHERE clienti.nr_client = tranzact.nr_client
                   AND tranzact.nr_fact = facturi.nr_fact
                   AND tranzact.tip = 'I'
                   AND privat = .T.
                SAVE TO TEMP clival2 ;
CLEAR
@0,0 SAY "De încasat de la neprivati : "
SELECT AVG(valoare) FROM clival1 ;
@5+1,0 SAY "De plătit celor privati : "
SELECT AVG(valoare) FROM clival2 ;
RETURN
```

Exercițiul 5.8

Care este media valorilor tranzacțiilor pe agenți? Dar pe clienți?

```
PROCEDURE 5_8
SELECT agenti.nr_agent , nume , valoare
        FROM agenti , tranzact , facturi
                WHERE agenti.nr_agent = tranzact.nr_agent
                   AND tranzact.nr_tranz = facturi.nr_tranz
                SAVE TO TEMP ageval ;
SELECT clienti.nr_client , denumire , valoare
        FROM clienti , tranzact , facturi
                WHERE clienti.nr_client = tranzact.nr_client
```


Scanned with OKEN Scanner

```
                AND tranzact.nr_tranz = facturi.nr_tranz
            SAVE TO TEMP clival ;
CLEAR
@0,0 SAY "Media pe agenti : "
SELECT nr_agent , nume , AVG(valoare)
        FROM ageval
            GROUP BY nr_agent , nume ;
@$+2,0 SAY "Media pe clienti : "
SELECT nr_client , denumire , AVG(valoare)
        FROM clival
            GROUP BY nr_client , denumire ;
RETURN
```

Exercițiul 5.9

Care este media valorilor cumpărărilor? Dar a vînzărilor?

```
PROCEDURE 5_9
SELECT tranzact.nr_tranz , tip , valoare
        FROM tranzact , facturi
            WHERE tranzact.nr_tranz = facturi.nr_tranz
        SAVE TO TEMP traval ;
CLEAR
@0,0 SAY "Media valorilor cumpărărilor : "
SELECT AVG(valoare)
        FROM traval
            WHERE tip = 'I' ;
@$+2,0 SAY "Media valorilor vînzărilor : "
SELECT AVG(valoare)
        FROM traval
            WHERE tip = 'E' ;
RETURN
```

Exercițiul 5.10

Care este suma cantităților produselor pe agenți și pe tip de tranzacție (cumpărare / vînzare)?

```
PROCEDURE 5_10
```


Scanned with OKEN Scanner

```
SELECT agenti.nr_agent , cant , tip
        FROM agenti , tranzact
                WHERE agenti.nr_agent = tranzact.nr_agent
        SAVE TO TEMP agemar ;
SELECT nr_agent , tip , SUM(cant)
        FROM agemar
                GROUP BY nr_agent , tip ;
RETURN
```

Exercițiul 6.4

Scrieți procedura de citire a datelor şi introducere în tabelul TRANZACT (cu comanda SQL *INSERT*), făcînd şi verificarea integrității de relație - unicitatea cheii primare - şi a integrității de referință.

Vom face toate verificările necesare înainte de INSERT (pentru a evita eroarea la execuție produsă de eventuala nerespectare a unicității cheii primare - asigurată prin intermediul indexului).

```
PROCEDURE 6_4
SET TALK off
CLEAR
SELECT * FROM tranzact ;
WAIT
CLEAR
a = 0              && sau STORE 0 TO a, b, c, d, e, g
b = 0
c = 0
d = 0
e = 0
f = DATE()
g = 0
h = 'I'
rasp = 'D'
DO WHILE .T.
        @15,0 SAY "Doriti terminarea ( D / N ) ?  : " GET rasp ;
                        FUNCTION "M D,N"
        READ
        IF rasp='D'
                EXIT
        ENDIF
```


Scanned with OKEN Scanner

```
@15,0 CLEAR
@0,0 SAY "Numărul de tranzactie : " GET a PICTURE "@Z 999999"
@1,0 SAY "Numărul de client :    " GET b PICTURE "@Z 99999999"
@2,0 SAY "Numărul de agent :     " GET c PICTURE "@Z 9999"
@3,0 SAY "Codul produsului :     " GET d PICTURE "@Z 999999"
@4,0 SAY "Cantitatea :           " GET e PICTURE "@Z 9999"
@5,0 SAY "Data tranzactiei :     " GET f PICTURE "@E" ;
                    DEFAULT DATE()
@6,0 SAY "Numărul facturii :     " GET g PICTURE "@Z 999999"
@7,0 SAY "Tip ( I / E ) :        " GET h FUNCTION "M I,E"
READ
unic   = 0
ex_cli = 0
ex_age = 0
&& presupunem că mai întîi se completează TRANZACT şi apoi FACTURI
SELECT COUNT(*) INTO unic
        FROM tranzact  WHERE nr_tranz = a ;
SELECT COUNT(*) INTO ex_cli
        FROM clienti    WHERE nr_client = b ;
SELECT COUNT(*) INTO ex_age
        FROM agenti    WHERE nr_agent = c ;
SELECT COUNT(*) INTO ex_cod
        FROM marfuri   WHERE cod_produs = d ;
* Atenţie: daca SELECT nu "găseşte" nimic, NU se creează variabila de memorie !
IF unic=0  .AND.  ex_cli=1  .AND.  ex_age=1
        INSERT INTO tranzact VALUES ( a , b , c , d , e , f , g , h ) ;
ELSE
        @20,0 SAY "Date inadmisibile - "
        @$,$+3 SAY CHR(7)                    && beep !
        DO CASE
            CASE unic > 0
                @$,$+3 SAY "Număr de tranzacţie deja ocupat ! "
            CASE ex_cli = 0
                @$,$+3 SAY "Client inexistent ! "
            CASE ex_age = 0
                @$,$+3 SAY "Agent inexistent ! "
            CASE ex_cod = 0
                @$,$+3 SAY "Marfa inexistenta ! "
              ENDCASE
ENDIF
WAIT ""
```


Scanned with OKEN Scanner

```
        @20,0 CLEAR
ENDDO
CLEAR
WAIT "Apăsaţi o tastă pentru a vedea conţinutul tabelului după actualizare ..."
SELECT * FROM tranzact ;
SET TALK on
RETURN
```

Exerciţiul 7.2

După "pornire" (adică 3.1):

a) Creaţi următoarele vizualizări:
- agenţii cu salariul >= 50 000 ce asigură măcar o tranzacţie;
- clienţii cu capital privat din Moldova;
- tranzacţiile din acest an;

b) Afişaţi elementele utile din catalogul relaţional.
c) Afişaţi toate liniile fiecărui tabel virtual creat.
d) Măriţi (în *view*) cu 20% salariul agenţilor ce au (strict) peste 50 000. Afişati toate liniile din tabelul de bază şi din vizualizare.
Atenţie: încercaţi ambele variante de SELECT-are (*ensembliste* / predicativă)!
e) Eliminaţi ce aţi creat la punctul a).

PROCEDURE 7_2

```
CREATE VIEW v1 AS
        SELECT * FROM agenti
                WHERE sal_baza >= 50000
                        AND nr_agent IN (
                                SELECT nr_agent FROM tranzact ) ;
CREATE VIEW v1bis AS
        SELECT DISTINCT agenti.nr_agent , nume , prenume , sal_baza
                FROM agenti , tranzact
                        WHERE sal_baza >= 50000
                                AND agenti.nr_agent = tranzact.nr_agent ;
CREATE VIEW v2 AS
        SELECT * FROM clienti
                WHERE privat = .T.
                  AND judet IN ( 'SV' , 'BT' , 'NT' , 'IS' , 'BC' , 'VS' , 'VN' , 'GL' ) ;
CREATE VIEW v3 AS
        SELECT * FROM tranzact
```


Scanned with OKEN Scanner

```
                WHERE YEAR(data_tranz) = YEAR(DATE()) ;
SELECT viewname , vcheck , readonly , sqltext FROM sysviews ;
SELECT * FROM sysvdps ;
WAIT "Conținutul primei vizualizări ..."
SELECT nr_agent , nume , prenume , sal_baza FROM v1 ;
WAIT "Varianta predicativă ..."
SELECT * FROM v1bis ;
WAIT "Clienții cu capital privat din Moldova ..."
SELECT * FROM v2 ;
WAIT "Tranzacțiile din acest an..."
SELECT * FROM v3 ;
UPDATE v1 SET sal_baza = 1.2 * sal_baza
        WHERE sal_baza > 50000 ;
WAIT "În tabelul de bază după actualizare ..."
SELECT nr_agent , nume , prenume , sal_baza FROM agenti ;
WAIT "În vizualizare ..."
SELECT nr_agent , nume , prenume , sal_baza FROM v1 ;
UPDATE v1bis SET sal_baza = 1.2 * sal_baza
        WHERE sal_baza < 120000 ;
* Rezultă un mesaj de eroare la compilare !
DROP VIEW v1 ;
DROP VIEW v1bis ;
DROP VIEW v2 ;
DROP VIEW v3 ;
RETURN
```

Exercițiul 7.3

Testați exemplul de vizualizare pentru integritatea de domeniu la AGENTI (inserînd linii în vizualizare!) .

```
PROCEDURE 7_3
CREATE VIEW vag AS
        SELECT * FROM agenti
                WHERE judet IN ( "SV" , "BT" , "NT" , "IS" , "BC" , "VS" )
                    AND sal_baza BETWEEN 40000 AND 200000
        WITH CHECK OPTION ;
INSERT INTO vag
        VALUES ( 100 , 'IONESCU' , 'ION' , 'Washington' , 'DC' ,
                    {2/2/90} , {3/3/60} , 'M' , .T. , 50000 , 25000 ) ;
```


Scanned with OKEN Scanner

```
SELECT * FROM vag ;
DROP VIEW vag ;
RETURN
```

Exercițiul 7.4

Creați o vizualizare cu agenții și tranzacțiile asigurate în acest an; creați apoi (din aceasta) un fișier permanent cu cei din Iași .

```
PROCEDURE 7_4
CLEAR
CREATE VIEW agetra AS
        SELECT agenti.nr_agent , nume , oras , nr_tranz , data_tranz
                FROM agenti , tranzact
                        WHERE agenti.nr_agent = tranzact.nr_agent
                            AND YEAR(data_tranz) = YEAR(DATE()) ;
SELECT * FROM agetra ;
SELECT * FROM agetra
        SAVE TO TEMP fis_at KEEP ;
DROP VIEW agetra ;
RETURN
```

Exercițiul 8.1

Scrieți o procedură care, folosind valoarea returnată de funcția *FKMAX()* (numărul de taste / combinații de taste ce pot fi programate), să determine - cu ajutorul funcției *FKLABEL(expN)* - corespondența între numele tastelor și valoarea numerică pentru comanda *SET FUNCTION* (de exemplu : F2 - 1).

```
PROCEDURE 8_1
SET TALK off
CLEAR
i = FKMAX()
k = 1    && valoarea 0 este asociată tastei F1 (neprogramabilă, fiind rezervată
         && pentru HELP)
DO WHILE k<=i
         IF k<21
                  @k-1,0 SAY STR(k,3) + "     " + FKLABEL(k)
         ELSE
```


Scanned with OKEN Scanner

```
            @k-21,0 SAY STR(k,3) + "    " + FKLABEL(k)
        ENDIF
        k = k + 1
        IF k=21
            WAIT "Se trece la următorul ecran ..."
            CLEAR
        ENDIF
ENDDO
SET TALK on
RETURN
```

Exercițiul 8.2

Scrieți un program pentru calcularea mediilor. Ca date de intrare aveți fișierul STUD.DBF cu structura:

nume	CHAR (15)
prenume	CHAR (15)
id	NUMERIC (3 , 0) && număr (matricol) de identificare
nota1	NUMERIC (3 , 0)
nota2 , nota3 , nota4	la fel

Nota 0 (zero) reprezintă absență la examenul respectiv. Ca rezultat se cere un fișier CLAS.DBF în care apar două cîmpuri suplimentare, *nr_crt* și *media*, articolele fiind sortate în ordinea descrescătoare a mediilor, apoi în ordinea crescătoare pentru nume, prenume și număr matricol.

Mediile se vor calcula într-o funcție, după următoarea regulă: pentru o singură absență se împarte suma notelor la 3, altfel la 4.

Înainte de a "porni", dați comenzile *USE STUD*, *LIST* și *USE* (altfel, rămîne deschis fișierul STUD.DBF). Atenție la literele mici / mari și eventualele spații anapoda de la începutul cîmpurilor de tip caracter!

După crearea CLAS.DBF, puteți încerca pe rînd *USE CLAS*, *EDIT* și apoi F2 (*BROWSE*).

> **Notă :** *nr_crt* este superfluu dacă am folosi fișierul creat numai din dBASE (există o funcție *RECCOUNT()*); îl creăm pentru a mai face o parcurgere a tabelului și pentru a vedea / lista direct liniile numerotate dintr-un utilitar (XTG, PCSHELL , NC ...).

PROCEDURE 8_2

Scanned with OKEN Scanner

```
SET TALK off
CREATE TABLE s1 (
        nume            CHAR(15) ,
        prenume         CHAR(15) ,
        id              NUMERIC(3,0) ,
        nota1           NUMERIC(3,0) ,
        nota2           NUMERIC(3,0) ,
        nota3           NUMERIC(3,0) ,
        nota4           NUMERIC(3,0) ) ;
LOAD DATA FROM stud INTO TABLE s1 ;
ALTER TABLE s1
        ADD ( nr_crt    NUMERIC(4,0) ,
              media     NUMERIC(4,2) ) ;
* Tabelul s1 putea fi creat şi de la început cu toate coloanele necesare.
UPDATE s1 SET nr_crt = 0 WHERE .T. ;
UPDATE s1 SET nume = LTRIM(UPPER(nume)) WHERE .T. ;
UPDATE s1 SET prenume = LTRIM(UPPER(prenume)) WHERE .T. ;
DECLARE c1 CURSOR
        FOR SELECT nota1 , nota2 , nota3 , nota4 , media FROM s1
                FOR UPDATE OF media ;
OPEN c1 ;
nrlinii = SQLCNT
DO WHILE nrlinii > 0
        FETCH c1 INTO x1 , x2 , x3 , x4 , x5 ;
        md = calc_med(x1,x2,x3,x4)
        UPDATE s1 SET media = md WHERE CURRENT OF c1 ;
        nrlinii = nrlinii -1
ENDDO
CLOSE c1 ;
* Cursorul este, oricum, închis automat după trecerea dincolo de ultima linie!
SELECT nr_crt , nume , prenume , id , nota1 , nota2 , nota3 , nota4 , media
        FROM s1
                ORDER BY media DESC , nume ASC , prenume ASC , id ASC
                        SAVE TO TEMP s2 ;
DROP TABLE s1 ;
DECLARE c2 CURSOR
        FOR SELECT nr_crt FROM s2
                FOR UPDATE OF nr_crt ;
OPEN c2 ;
nrlinii = SQLCNT
i = 1
```


Scanned with OKEN Scanner

```
DO WHILE nrlinii > 0
        FETCH c2 INTO x1 ;
        UPDATE s2 SET nr_crt = i WHERE CURRENT OF c2 ;
        nrlinii = nrlinii -1
        i = i + 1
ENDDO
CLOSE c2 ;
SELECT * FROM s2 SAVE TO TEMP clas KEEP ;
SET TALK on
RETURN

FUNCTION calc_med
PARAMETERS n1 , n2 , n3 , n4
nrabs = 0
IF n1=0                          && se numără cîte absenţe sînt
        nrabs = nrabs + 1
ENDIF
IF n2=0
        nrabs = nrabs + 1
ENDIF
IF n3=0
        nrabs = nrabs + 1
ENDIF
IF n4=0
        nrabs = nrabs + 1
ENDIF
IF nrabs = 1
        RETURN ( n1 + n2 + n3 + n4 ) / 3
ELSE
        RETURN ( n1 + n2 + n3 + n4 ) / 4
ENDIF
```

Exerciţiul 8.3

Scrieţi un (embrion de) program pentru calcularea salariilor agenţilor comerciali pentru luna precedentă celei curente .

La salariul de bază se adaugă 0.3% din tranzacţiile "cumpărare" şi respectiv 0.6% din tranzacţiile "vînzare" intermediate.

În fişierul DATEPERS.DBF aveţi (pentru agenţi) următoarele elemente:

Scanned with OKEN Scanner

```
nume        CHAR(10)
prenume     CHAR(10)
nr_agent    NUMERIC(4,0)
sporuri     NUMERIC(2,0)  && procente din bază + comision
impozit     NUMERIC(2,0)  && procente din total brut
retineri    NUMERIC(6,0)  && în cifre absolute
```

Ca rezultat, se cere un fişier NETTO.DBF (tot cu o coloană *nr_crt*) în care să apară liniile sortate crescător după *nume, prenume, nr_agent*.

PROCEDURE 8_3

```
SET TALK off
SELECT agenti.nr_agent , tip , valoare , data_tranz , .T.
        FROM agenti , tranzact , facturi
                WHERE agenti.nr_agent = tranzact.nr_agent
                   AND tranzact.nr_tranz = facturi.nr_tranz
        SAVE TO TEMP tab0 ( nr_agent , tip , valoare , data_tranz , luna_prec ) ;
* Atenţie: NU se admit UDF (User Defined Functions) în interogările SQL !

DECLARE c0 CURSOR
        FOR SELECT data_tranz , luna_prec FROM tab0
                FOR UPDATE OF luna_prec ;
OPEN c0 ;
nrlinii = SQLCNT
DO WHILE nrlinii > 0
        FETCH c0 INTO alfa , beta ;
        IF testluna(alfa) = .F.
                UPDATE tab0 SET luna_prec = .F. WHERE CURRENT OF c0 ;
        ENDIF
        nrlinii = nrlinii - 1
ENDDO
CLOSE c0 ;

SELECT nr_agent , tip , valoare
        FROM tab0
                WHERE luna_prec = .T.
                        SAVE TO TEMP tab1 ;

* se obţin numai datele pentru luna precedentă ;

SELECT nr_agent , tip , SUM(valoare) , 0.
```


Scanned with OKEN Scanner

```
        FROM tab1
                GROUP BY nr_agent , tip
                        SAVE TO TEMP tab2 ( nr_agent , tip , valo , com ) ;
```

* Există şi funcţia dBASE *VAL()*.

```
UPDATE tab2 SET com = 3/1000 * valo WHERE tip = 'I' ;
UPDATE tab2 SET com = 6/1000 * valo WHERE tip = 'E' ;

SELECT nr_agent , SUM(com)
        FROM tab2
                GROUP BY nr_agent
                        SAVE TO TEMP tab3 ( nr_agent , cms ) ;
```

* tab3 este creat cu coloana *cms* de tip NUMERIC(20.2) chiar după
* *SET DECIMALS TO 0* !

```
SELECT nr_agent , sal_baza , 0
        FROM agenti
                WHERE nr_agent NOT IN ( SELECT nr_agent FROM tab3 )
                        SAVE TO TEMP xxx ( nr_agent , sal_baza , comis ) ;
SELECT agenti.nr_agent , sal_baza , cms
        FROM agenti , tab3
                WHERE agenti.nr_agent = tab3.nr_agent
                        SAVE TO TEMP yyy ( nr_agent , sal_baza , comis ) ;
SELECT * FROM xxx
        UNION
                SELECT * FROM yyy
                        SAVE TO TEMP tab4 ;

CREATE TABLE date_ag (
        nume            CHAR(10) ,
        prenume         CHAR(10) ,
        nr_agent        NUMERIC(4,0) ,
        sporuri         NUMERIC(2,0) ,
        impozit         NUMERIC(2,0) ,
        retineri        NUMERIC(6,0) ) ;
LOAD DATA FROM datepers INTO TABLE date_ag ;

SELECT 0 , tab4.nr_agent , nume , prenume , sal_baza , comis ,
          sporuri , impozit , retineri , 0
```


Scanned with OKEN Scanner

```
        FROM tab4 , date_ag
                WHERE tab4.nr_agent = date_ag.nr_agent
        SAVE TO TEMP tab5 ( nr_crt, nr_agent, nume, prenume, sal_baza,
                                comision, sporuri, impozit, retineri, sal_net) ;
* implicit , face nr_crt ca NUMERIC(20,2)
DROP TABLE date_ag ;
* tabelele temporare se şterg oricum (dacă nu se indică KEEP):
CREATE TABLE tab5a (
        nr_crt          NUMERIC(4,0) ,
        nr_agent        NUMERIC(4,0) ,
        nume            CHAR(10) ,
        prenume         CHAR(10) ,
        sal_baza        NUMERIC(6,0) ,
        comision        NUMERIC(6,0) ,
        sporuri         NUMERIC(2,0) ,
        impozit         NUMERIC(2,0) ,
        retineri        NUMERIC(6,0) ,
        sal_net         NUMERIC(6,0) ) ;
INSERT INTO TABLE tab5a SELECT * FROM tab5 ;
UPDATE tab5a
   SET sal_net = (sal_baza + comision)*(1+sporuri/100)*(1-impozit/100) - retineri
        WHERE .T. ;
SELECT * FROM tab5a
        ORDER BY nume ASC , prenume ASC , nr_agent ASC
                SAVE TO TEMP tab6 ;
DROP TABLE tab5a ;

DECLARE c1 CURSOR
        FOR SELECT nr_crt FROM tab6
                FOR UPDATE OF nr_crt ;
OPEN c1 ;
nrlinii = SQLCNT
i = 1
DO WHILE nrlinii > 0
        FETCH c1 INTO alfa ;
        UPDATE tab6 SET nr_crt = i WHERE CURRENT OF c1 ;
        nrlinii = nrlinii - 1
        i = i + 1
ENDDO
CLOSE c1 ;
UNLOAD DATA TO netto FROM TABLE tab6 ;
```


Scanned with OKEN Scanner

```
SET TALK on
RETURN

FUNCTION testluna
PARAMETERS dt
* Returnează .T. dacă luna calendaristică corespunzătoare expD transmisă ca
* parametru este luna imediat anterioară celei curente.
l = MONTH(DATE())
a = YEAR(DATE())
ll = MONTH(dt)
al = YEAR(dt)
IF l>1 .AND. a=al .AND. ll=l-1
        RETURN .T.
ELSE
        IF l=1 .AND. al=a-1 .AND. ll=12
                RETURN .T.
        ELSE
                RETURN .F.
        ENDIF
ENDIF
```

Referințe bibliografice pentru capitolul de dBase

[1] Miranda, S.M., *L'art des bases de données (3) Comprendre et évaluer SQL*, Eyrolles, Paris, 1990

[2] ***, *Ghid de utilizare dBASE IV*, Microinformatica, Cluj, 1991

[3] ***, *dBASE IV Utilizarea limbajului de comandă*, ROMSOFT, București, 1992

Scanned with OKEN Scanner

Cuprins

Introducere

I. Programarea în C

A. Probleme din îndrumarul de laborator

Problema 3.3.....5
Problema 3.4.....6
Problema 3.5.....8
Problema 4.1.....9
Problema 4.2.....17
Problema 4.3.....18
Problema 6.3.....20
Problema 7.1.....21
Problema 7.2.....25
Problema 7.3.....28
Problema 8.1.....32
Problema 8.2.....33
Problema 8.3.....34
Problema 9.1.....37
Problema 9.2.....40
Problema 10.2.....45
Problema 11.3.....48
Problema 12.2.....49
Problema 12.4.....50
Problema 13.2.....51
Problema 13.3.....53
Problema 13.4.....54
Problema 14.1.....59
Problema 14.2.....61
Problema 14.3.....63
Problema 15.3.....67
Problema 16.2.....69
Problema 16.3.....71
Problema 16.5.....71
Problema 18.1.....73
Problema 18.2.....74
Problema 18.3.....77
Problema 19.1.....81
Problema 19.2.....85
Problema 19.3.....88
Problema 19.4.....93
Problema 19.7.....96
Problema 21.1.....97
Problema 21.4.....99
Problema 21.6.....101
Problema 22.1.....103
Problema 22.2.....104
Problema 22.3.....105
Problema 24.1.....107
Problema 24.2.....107
Problema 24.4.....109
Problema 25.1.....111
Problema 25.4.....112
Problema 26.1.....114
Problema 26.2.....117
Problema 27.3.....121
Problema 28.2.....126

B. Probleme suplimentare

B1.....128
B2.....129
B3.....133
B4.....134

Scanned with OKEN Scanner

B5.....134
B6.....135
B7.....137
B8.....137
B9.....140
B10.....141
B11.....142
B12.....144
B13.....145
B14.....149
B15.....151
B16.....152
B17.....153
B18.....158
B19.....161
B20.....163
B21.....166
B22.....170
B23.....173
B24.....175
B25.....176
B26.....178
B27.....180
B28.....183
B29.....184
B30.....185
B31.....186
B32.....187
B33.....189

Referințe bibliografice.....191

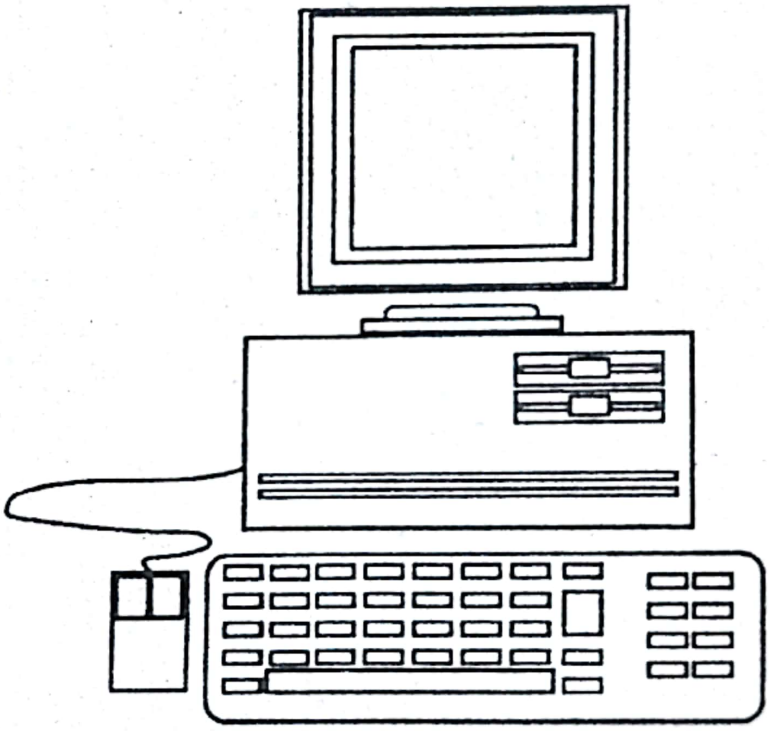
II. Baze de date - dBase IV

Exercițiul 2.2-3.1.....192
Exercițiul 3.3.....195
Exercițiul 3.4.....195
Exercițiul 3.5.....196
Exercițiul 3.6.....196
Exercițiul 3.7.....197
Exercițiul 4.3.....198
Exercițiul 4.4.....198
Exercițiul 4.5.....199
Exercițiul 4.6.....199
Exercițiul 4.7.....199
Exercițiul 4.8.....200
Exercițiul 4.9.....200
Exercițiul 4.10.....201
Exercițiul 4.11.....202
Exercițiul 5.1.....202
Exercițiul 5.2.....203
Exercițiul 5.3.....204
Exercițiul 5.5.....205
Exercițiul 5.6.....207
Exercițiul 5.7.....208
Exercițiul 5.8.....208
Exercițiul 5.9.....209
Exercițiul 5.10.....209
Exercițiul 6.4.....210
Exercițiul 7.2.....212
Exercițiul 7.3.....213
Exercițiul 7.4.....214
Exercițiul 8.1.....214
Exercițiul 8.2.....215
Exercițiul 8.3.....217

Referințe bibliografice.....221

Scanned with OKEN Scanner

Editura LORELEY
IAŞI
1996



ISBN 973-96962-6-0

Scanned with OKEN Scanner